NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A STUDY OF THE SEASTAR UNDERWATER ACOUSTIC LOCAL AREA NETWORK CONCEPT

by

Bjørn E. A. Kerstens

December 2007

Thesis Advisor:	Joseph A. Rice
Co-Advisor:	Lawrence J. Ziomek

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 2315 Jefferson Davis Highway, Suite 2304, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A Study of the Seastar Underwater Acoustic Local Area Network Concept		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Bjørn E. A. kerstens			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
23a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**23b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

This research considers the "Seastar" concept of an underwater local-area network (LAN) having a central node and multiple peripheral nodes. The concept of operation for the Seastar LAN involves the delivery of large volumes of digital information from the peripheral nodes through direct acoustic communication links to a sophisticated central node for assimilation (e.g., beamforming, fusion). For a design range of 500 meters, link budget analysis in combination with parametric analysis evaluates physical-layer parameters including optimum carrier frequency, spectral bandwidth, modulation techniques, achievable bit rate, and energy budget. Performance data obtained from a prototype Seastar LAN constructed from existing acoustic modems guided the creation of a Seastar numerical simulation. Monte Carlo simulation studies examine the relative merits of networking strategies such as TDMA polling and token-based TDMA. Seastar is shown to meet the anticipated requirements for undersea LAN applications such as sensor networks, undersea vehicle swarms, and dive teams.

14. SUBJECT TERMS Acoustics, sound, ocean, acoustic communications, underwater networks, LAN, network simulation, Seaweb, Seastar			**15. NUMBER OF PAGES** 177
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A STUDY OF THE SEASTAR UNDERWATER ACOUSTIC LOCAL AREA NETWORK CONCEPT

Bjørn E. A. Kerstens
Lieutenant, Royal Netherlands Navy

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
December 2007

Author: Bjørn E. A. Kerstens

Approved by: Joseph A. Rice
Thesis Advisor

Dr. Lawrence J. Ziomek
Co-Advisor

Dr. Kevin Smith
Chairman, Engineering Acoustics Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research considers the “Seastar” concept of an underwater local-area network (LAN) having a central node and multiple peripheral nodes. The concept of operation for the Seastar LAN involves the delivery of large volumes of digital information from the peripheral nodes through direct acoustic communication links to a sophisticated central node for assimilation (e.g., beamforming, fusion). For a design range of 500 meters, link budget analysis in combination with parametric analysis evaluates physical-layer parameters including optimum carrier frequency, spectral bandwidth, modulation techniques, achievable bit rate, and energy budget. Performance data obtained from a prototype Seastar LAN constructed from existing acoustic modems guided the creation of a Seastar numerical simulation. Monte Carlo simulation studies examine the relative merits of networking strategies such as TDMA polling and token-based TDMA. Seastar is shown to meet the anticipated requirements for undersea LAN applications such as sensor networks, undersea vehicle swarms, and dive teams.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

As my study and life in the “New World” comes to an end, I realize that I have had a great opportunity to develop myself and become a more complete person and officer. This experience would never have been possible without the support of many people whom I wish to thank:

To my wife and best friend, Brenda, for her support and care, and to my children, Jules and Midas, for ensuring that I had a life beyond the books.

To my parents, for providing the discipline required to fulfill this education.

To my thesis advisor, Joe Rice, for giving me this great opportunity, the support and the freedom that I was hoping to find in my thesis research.

To my co-advisor, Professor Lawrence Ziomek, for his contributions and the time spent on issues that were beyond my knowledge.

To Dana Hesse from the Office of Naval Research, for sponsoring this research.

To the staff at SPAWAR Systems Center in San Diego: especially Chris Fletcher and Bob Creber for sharing their experience with me to understand the practical implications of underwater networks and for providing me with the tools to create a prototype network. I would also like to thank Bill Marn for his support throughout AUV Fest 2007 and for being a source of inspiration for the leadership required to run an experiment successfully.

To Bob Broadston from the ECE department for his discussions on network theory that provided me with useful ideas for shaping Seastar.

To Professor Steve Baker for his practical knowledge on transducers and his experience and guidance in performing acoustic measurements in the anechoic chamber and to Jay Adeff for providing me with the right equipment.

To Sam Barone, not only for his technical contribution, but especially for his mental support and necessary distraction from working in the anechoic chamber.

To Dale Green and Rob Pinelli from Teledyne Benthos, for sharing their knowledge on the Benthos modems.

To Lance Presnall of Brüel and Kjær, for his advice in using the PULSE analyzer.

To the faculty and staff of the NPS physics department, for making me enthusiastic in the field of physics and broadening my technical vision.

To the instructors of the sensor, weapon and command systems section of the Netherlands Defense Academy's Faculty of Military Sciences, for preparing me for this study. A special word of thanks to Vincent van Leijen for assisting me and keeping me informed on developments in the Royal Netherlands Navy and Professor Frans Absil for providing the conditions that made my study at NPS a success.

I. INTRODUCTION

The Seaweb wide-area network (WAN) concept [1] provides for local-area networks (LANs) having a sophisticated central node that collects and fuses undersea data from a set of relatively simple peripheral nodes, as illustrated in Figure 1.

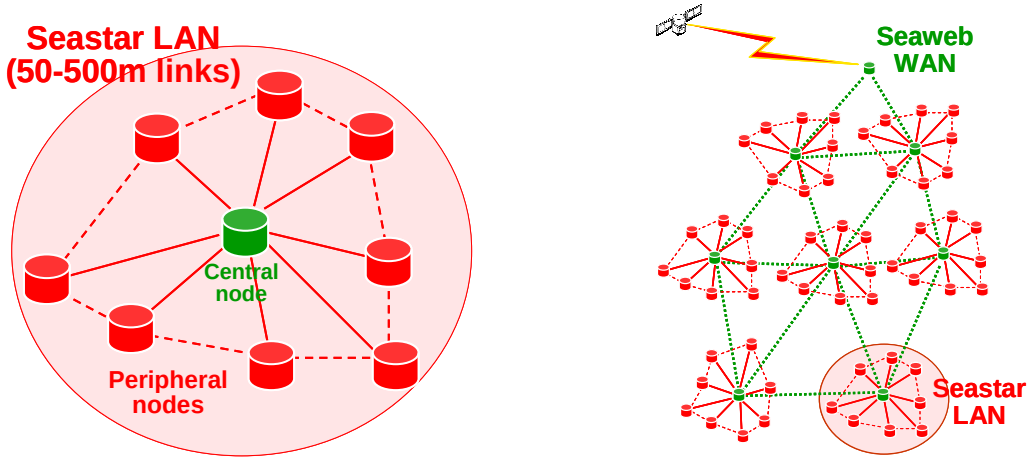

Figure 1 Left: The Seastar concept involves asymmetric, centralized topologies. Relatively simple peripheral nodes (red) report time-series data at high-bit-rates to sophisticated central nodes (green), where data fusion is performed for the local area. Peripheral nodes may receive low-bit-rate utility packets from the central node and from their peers. Right: Wide-area communications (green links) between central nodes and theater communications through gateway nodes occur via Seaweb networking in a lower band of the acoustic spectrum.

In more general terms, localized clusters of nodes (e.g., sensors, crawlers, divers) assimilate as subnets through the formation of LANs, each containing a central node and peripheral nodes distributed in an undersea region up to one square kilometer (km^2). The Seaweb LAN with centralized, or star, topology is called “Seastar” and is motivated by the desire for an additional tier of local-area communications compatible with and complementary to Seaweb wide-area acoustic communications. The Seastar tier uses a higher-frequency portion of the acoustic spectrum made possible by shorter communication ranges and necessitated by the high throughput of the LAN. The baseline Seastar topology is centralized, with axial asymmetry and peer-to-peer capability.

A. SCOPE

This thesis explores candidate Seastar networking strategies and considers physical-layer and link-layer attributes. The effects of the underwater communications channel on carrier frequency, signal bandwidth, capacity and energy budget are studied and tradeoffs regarding modulation type, access method and topology are presented. In order to investigate the feasibility of an underwater acoustic LAN, a Seastar prototype was developed and tests were conducted both in air and in water. The experimental results provide useful information for design purposes and form the basis of a network simulation that was developed as part of this research. The simulation is used to study different network strategies under various conditions and to perform case studies that are included in this document. In summary, this thesis provides tradeoffs that allow both designers and operational users to validate the possibilities and limitations of the Seastar concept.

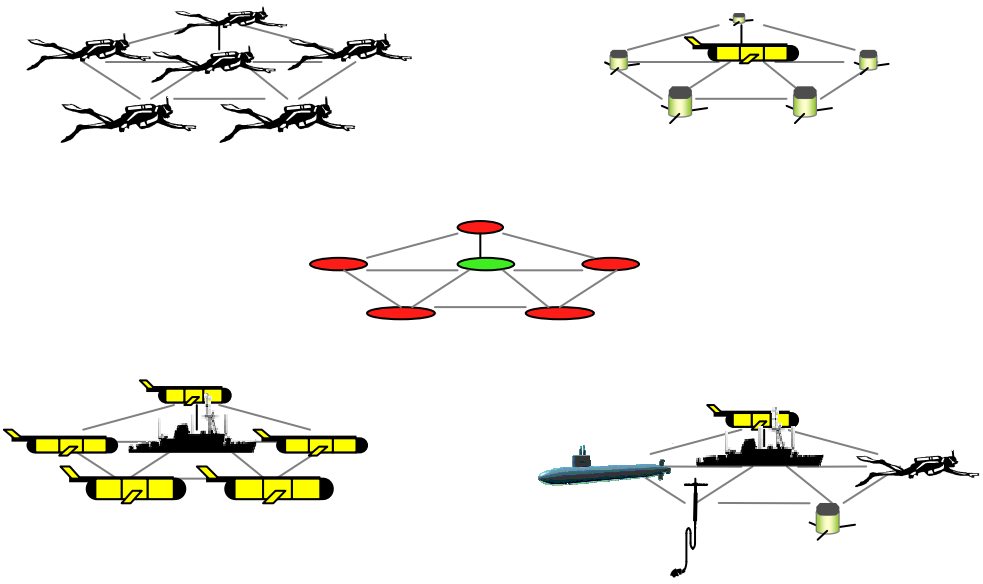

Figure 2 Seastar applications include sensor arrays, sensor clusters, unmanned undersea vehicle formations, and dive teams.

The broad scope of the research topic requires judicious restrictions in the research that was conducted. Throughout this study a fixed communication range of 500 meters (m) is considered. Propagation is generally assumed to occur along a direct path in

a homogeneous (constant speed of sound) environment. The use of mobile communication nodes is excluded in simulations and experiments but included in discussions. Issues such as interference between adjacent Seastar clusters and integration of Seastar and Seaweb need to be studied, but are beyond the scope of this thesis.

B. APPROACH AND STRUCTURE

The topic of acoustic communications brings two communities together (communication theory and underwater acoustics), each using their own language. The literature on this topic is not always consistent in translating the terminology of the communications discipline and the acoustics discipline.

Chapter II adopts the conventions of the underwater acoustician to discuss the physical properties of the underwater communications channel, and derive an expression for the acoustic signal-to-noise power ratio and corresponding sonar equation. A link budget analysis is performed for conditions under which Seastar is anticipated to operate with the purpose of determining an optimum carrier frequency.

Chapter III discusses the physical layer of Seastar and contains a case study to provide the reader with some hardware, bandwidth, capacity and energy budget considerations. The central part of Chapter III contains a study on modulation techniques.

Chapter IV brings us deeper into network theory and suggests several network topologies. It discusses variations in access to the medium and studies control of information flow. The chapter ends with an outline of experiments that resulted in the development of the first Seastar prototype.

Chapter V leads the reader further into the prototype analysis by providing an overview of network performance results as obtained during an in-water deployment in St. Andrews Bay, FL. The results of these experiments were used to develop a network simulation tool that provides us with data regarding the performance of other network strategies that could not easily be implemented and tested with existing hardware.

Chapters VI and VII show the setup of the simulation and display parametric results allowing a better view on Seastar possibilities and limitations.

Finally, since an optimal Seastar strategy depends on the networks' purpose, three case studies are performed to provide operational end-users with some ideas that may help shape future applications for Seastar. These case studies illustrate the conclusions stated in Chapter VIII.

II. THE COMMUNICATION CHANNEL

An important characteristic of the underwater acoustic communication channel is the dependence of path loss and ambient noise on communication distance and transmission frequency. This, therefore, affects other communication parameters like signal bandwidth and data rate. In this chapter we look at the influence that the environment has on the signal that is transmitted and from this we will determine the optimum carrier frequency and bandwidth to use. We will limit our analytical scope to path loss and ambient noise, but the effects of multi-path propagation and single-path fluctuations will be briefly discussed when analyzing appropriate waveforms.

A. THE PHYSICAL CHANNEL

The underwater communication channel can be described as a cylindrical waveguide that is bounded by the sea surface and the sea floor with communication ranges generally exceeding water depth. Variations in channel composition, depth and temperature cause refraction of sound, often combined with surface and bottom reflections and convolutions with imparted boundary conditions, resulting in a multitude of possible propagation paths for communication signals. Each single path imposes temporal, spatial and frequency-dependent amplitude and phase fluctuations on a waveform. The accumulation of these paths at the receiver produces multi-path reception replete with distortion, and dispersion of the waveform. These deleterious effects translate into fading, inter-symbol interference, and loss of coherence.

B. ACOUSTIC SIGNAL-TO-NOISE POWER RATIO

The sometimes inconsistent use of terminology in the interdisciplinary field of underwater acoustic communications by the technical communications community [2–4] and underwater acoustics community [5–7] creates confusion, and forms the motivation for deriving some important expressions in generally accepted acoustic terms. This will be done in accordance with [8–9]. To analyze the communication losses through the

channel we will introduce the term acoustic signal-to-noise power ratio SNR_a at the receiver which is defined as follows [9]:

$$SNR_a \triangleq \frac{E\left\{\left|p_s\left(t,\mathbf{r}_1\right)\right|^2\right\}}{E\left\{\left|p_n\left(t,\mathbf{r}_1\right)\right|^2\right\}} \tag{2.1}$$

where $p_s(t,\mathbf{r}_1)$ and $p_n(t,\mathbf{r}_1)$ are the acoustic pressures due to signal and noise, respectively, incident upon a receiver located at $\mathbf{r}_1=(x_1,y_1,z_1)$, and $E\{\bullet\}$ is the expected value. The SNR_a informs us on the status of the transmission at the input to the receiver before the signal is processed.

1. Signal

A communications signal in an underwater acoustic channel usually consists of a band of frequencies. Therefore, in order to derive an equation for SNR_a that is equivalent to the "narrowband" signal-to-noise ratio equation in [2], the time-harmonic acoustic pressure at time t in seconds (s) and range $r_{0,1}$ in meters (m) from the source (see Figure 3) can be expressed as:

$$p_s\left(t,\mathbf{r}_1\right)=\left|p_{f,s}\left(\mathbf{r}_1\right)\right|\cos\left(2\pi ft+\angle p_{f,s}\left(\mathbf{r}_1\right)\right) \tag{2.2}$$

where, (e.g. see [10])

$$p_{f,s}\left(\mathbf{r}_1\right)=p_0\frac{R}{r_{0,1}}e^{-\alpha(f)\left(r_{0,1}-R\right)}e^{-jk\left(r_{0,1}-R\right)}, \tag{2.3}$$

$$p_0=\left|p_0\right|e^{+j\angle p_0} \tag{2.4}$$

is the spatial-dependent part of the time-harmonic acoustic pressure in Pascals (Pa) at one meter from the source and $\alpha(f)$ is the frequency-dependent attenuation coefficient in Nepers per meter (Np/m). We assume that the signal sound source is a motionless, time-harmonic, omnidirectional point source and that the medium is unbounded and viscous with a constant speed of sound, and R=1 m is the reference range from the source.

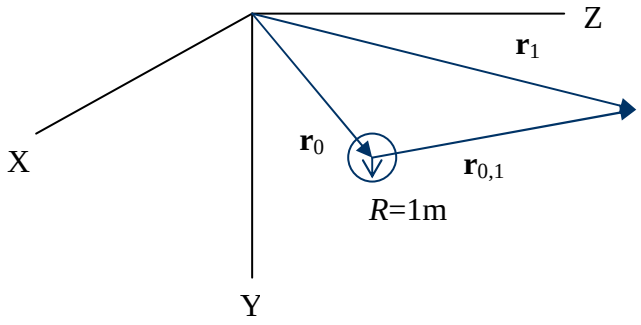

Figure 3 Source ($\mathbf{r}_0$) – receiver ($\mathbf{r}_1$) geometry where R = 1 m is the reference range from the source.

Substitution of $P_0=|p_0|$ into (2.4) and computing the magnitude of (2.3) yields

$$\left|p_{f,s}\left(\mathbf{r}_1\right)\right| = P_0 \frac{R}{r_{0,1}} e^{-\alpha(f)\left(r_{0,1}-R\right)}. \tag{2.5}$$

If we treat $p_s(t,\mathbf{r}_1)$ as being deterministic and since it is also periodic, we can replace the numerator of (2.1) with the time-average power

$$\left\langle \left|p_s\left(t,\mathbf{r}_1\right)\right|^2 \right\rangle = \frac{1}{T}\int_0^T \left|p_s\left(t,\mathbf{r}_1\right)\right|^2 dt = \frac{1}{2}\left|p_{f,s}\left(\mathbf{r}_1\right)\right|^2. \tag{2.6}$$

Substituting (2.5) into (2.6) gives the following equation:

$$\left\langle \left|p_s\left(t,\mathbf{r}_1\right)\right|^2 \right\rangle = \left(\frac{\sqrt{2}}{2} P_0 \frac{R}{r_{0,1}}\right)^2 e^{-2\alpha(f)\left(r_{0,1}-R\right)} = p_{rms,s}^2\left(\mathbf{r}_1\right) \tag{2.7}$$

where $p_{rms,s}(\mathbf{r}_1)$ is the root-mean square value of the acoustic pressure of the signal in Pa at a receiver at position $\mathbf{r}_1$ that was transmitted omnidirectionally by a source at position $\mathbf{r}_0$.

2. Noise

Noise in the ocean is generated by various sources such as wind, shipping and flow noise. To come up with a general expression for the average power of the acoustic pressure that is created by the noise and received by the receiver at $\mathbf{r}_1$, we assume that

$p_n(t,\mathbf{r}_1)$ is a zero mean, wide-sense stationary (WSS), random process, representing an arbitrary function of time. As a result, the autocorrelation function of the noise can be expressed as follows:

$$R_{p_n}(t,t',\mathbf{r}_1) = R_{p_n}(t-t',\mathbf{r}_1) = R_{p_n}(\Delta t,\mathbf{r}_1) = E\left\{p_n(t,\mathbf{r}_1)\,p_n^*(t',\mathbf{r}_1)\right\} \tag{2.8}$$

Since $R_{p_n}(\Delta t,\mathbf{r}_1)$ forms a Fourier transform pair with its power spectral density function $S_{p_n}(\eta,\mathbf{r}_1)$ in Pa^2/Hz, the average power of the zero mean, WSS noise is given by the following relation:

$$\sigma_{p_n}^2(\mathbf{r}_1) = R_{p_n}(0) = E\left\{\left|p_n(t,\mathbf{r}_1)\right|^2\right\} = \int_{-\infty}^{\infty} S_{p_n}(\eta,\mathbf{r}_1)d\eta \tag{2.9}$$

where $\sigma_{p_n}^2(\mathbf{r}_1)$ is the noise variance and η represents frequency in hertz (Hz). If we consider a limited noise bandwidth of Δf centered around frequency f, and taking into account that the power spectrum is an even function of frequency, we obtain the following expression for the denominator of (2.1):

$$\sigma_{p_n}^2(\mathbf{r}_1) = E\left\{\left|p_n(t,\mathbf{r}_1)\right|^2\right\} = 2\int_{f-(\Delta f/2)}^{f+(\Delta f/2)} S_{p_n}(\eta,\mathbf{r}_1)d\eta = 2S_{p_n}(f,\mathbf{r}_1)\Delta f\ . \tag{2.10}$$

This general expression for the average power of the noise can easily be converted to a noise level by letting $\Delta f = 1\ Hz$. Substituting (2.7) into the numerator of (2.1) and (2.9) into the denominator of (2.1) yields:

$$SNR_a = \frac{p_{rms,s}^2(\mathbf{r}_1)}{\sigma_{p_n}^2(\mathbf{r}_1)}\left(\frac{P_{ref}^2}{P_{ref}^2}\right) = \frac{\left(\frac{\sqrt{2}P_0/2}{P_{ref}}\frac{R}{r_{0,1}}\right)^2}{\left(\frac{2S_{p_n}(f,\mathbf{r}_1)\Delta f}{P_{ref}^2}\right)}e^{-2\alpha(f)(r_{0,1}-R)} \tag{2.11}$$

where P_{ref} =1 μPa is the root-mean-square reference pressure amplitude.

The last step is to express (2.11) in decibels (dB) and set the reference range R to 1 m. Doing so yields an expression for SNR_a (in dB) as shown in (2.12) in more familiar terms; namely, source level (*SL*), noise level (*NL*) and transmission loss (*TL*), where transmission loss consists of a spherical spreading term and a frequency-dependent attenuation term:

$$SNR_a(\text{in dB}) = SL - NL - TL \tag{2.12}$$

where

$$SNR_a\left(\text{in dB}\right) = 10\log_{10} SNR_a \tag{2.13}$$

$$SL = 20\log_{10}\left(\frac{\sqrt{2}P_0/2}{P_{ref}}\right) \quad \text{dB re } P_{ref} \tag{2.14}$$

$$NL = 10\log_{10}\left(\frac{2S_{p_n}(f,\mathbf{r}_1)\Delta f}{P_{ref}^2}\right) \quad \text{dB re } P_{ref}^2 \tag{2.15}$$

$$TL = 20\log_{10}(r_{0,1}) + \alpha'(f)(r_{0,1}-1) \quad \text{dB re } P_{ref} \tag{2.16}$$

and $\alpha'(f) = 8.686\alpha(f)$ where $\alpha'(f)$ is in dB/m and $\alpha(f)$ is in Np/m. We will use analytical expressions based on empirical data for $\alpha'(f)$ and $S_{p_n}(f,\mathbf{r}_1)$, in order to allow us to perform the next step in analyzing underwater acoustic communications. We will do this by performing a link budget analysis, a method that was described by Hansen [11] and proved to be viable in the underwater environment.

C. LINK BUDGET ANALYSIS

1. Transmission Loss

Spherical spreading and frequency-dependent attenuation are the two components of transmission loss (*TL*) in (2.16). Some of the literature (e.g., see [1]) expresses the spreading component $20\log_{10}(r_{0,1})$ in the form of $k10\log_{10}(r_{0,1})$, $1 \le k \le 2$, where $k = 1$ is used for cylindrical spreading, $k = 2$ for spherical spreading and $k = 1.5$ for the so called practical spreading. This may result in severe under- or overestimation. Although it is recognized that physics-based modeling of multi-path propagation is more realistic and provides a more accurate representation of the losses in the channel, the choice has been made to use the spherical spreading direct path model.

Various empirical formulas for the attenuation coefficient $\alpha'(f)$ can be found in the open literature. The empirical formula provided by Francois and Garrison [12, 13] is claimed by them to apply to all oceanic conditions and frequencies from 200 Hz to 1 megahertz (MHz) with an estimated accuracy of 5% and appears to embrace and improve

upon the studies conducted by Thorp [14], Fisher and Simmons [15], and Marsh and Schulkin [16]. Since we anticipate using carrier frequencies between 20 and 100 kilohertz (kHz), we will use the formula in Francois and Garrison [12, 13] for the attenuation coefficient. Figure 4 shows two examples of frequency dependency of the attenuation coefficient expressed in dB/km for two different temperatures, $T = 14\ C^{\circ}$ and $T = 20\ C^{\circ}$, salinity $S = 35$ parts per thousand (ppt), acidity $pH = 8.0$ and depth $D = 50$ m.

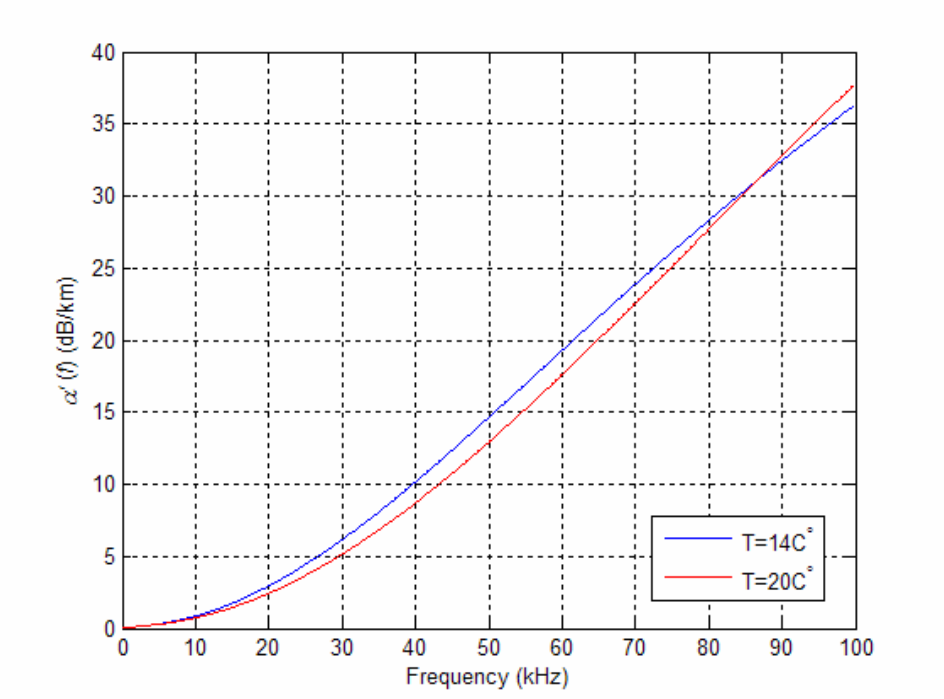

Figure 4 Attenuation coefficient $\alpha'(f)$ in dB/km versus frequency in kHz based on Francois and Garrison [12, 13] for the above described conditions.

It is clear that the attenuation coefficient increases with frequency, but since we consider a maximum transmission range of 500 m for Seastar, the question arises at what frequency will the attenuation coefficient start to gain influence compared to spherical spreading.

Figure 5 indicates that the impact of frequency-dependent attenuation at 500 m is insignificant up to approximately 25 kHz, but adds 7 dB to the *TL* at 50 kHz. Beyond 50 kHz the influence of attenuation increases quickly. Increasing the carrier frequency and signal bandwidth therefore comes at the cost of range reduction.

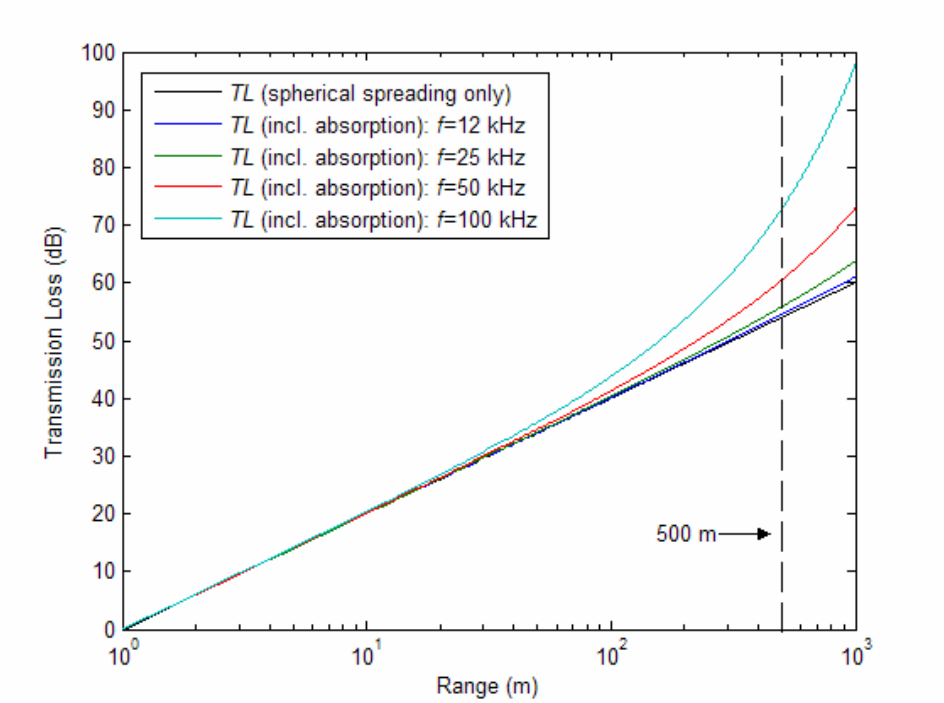

Figure 5 Relative influence of losses caused by frequency-dependent attenuation on total transmission loss in dB (vertical axis) for different ranges in m (horizontal axis). For a range of 500 m, losses due to attenuation are of minor importance for frequencies below 25 kHz compared to losses due to spherical spreading.

2. Noise Level

The next factor in (2.12) that needs to be addressed is the noise level given by (2.15). If we consider future Seastar deployment without defining geographic restrictions we learn [5, 6, 17] that underwater noise in general contains several contributors that each affect different frequency bands and that can be described empirically. Comparisons for different situations [6] show that different oceanographic conditions, such as water depth

and temperature, do influence the appearance of noise but that the empirical formulas [5] satisfy a description for noise in the most general case. Noise measurements are also reported as noise spectrum levels (*NSL*) in dB with reference to 1 μPa^2/Hz (e.g., see [10])

$$NSL = 10\log_{10}\left(\frac{2S_{p_n}(f,\mathbf{r}_1)}{\mu\text{Pa}^2/\text{Hz}}\right). \tag{2.17}$$

The *NSL* in dB based on empirical formulas by Coates [5] is generally dependent on four sources dominating certain frequency bands, namely turbulence (<10 Hz), shipping (10-200 Hz), wind (0.2-100 kHz) and thermal activity (>100 kHz). Figure 6 indicates that the *NSL* for the frequency band between 1-100 kHz that is typically used for underwater communications is mainly due to wind.

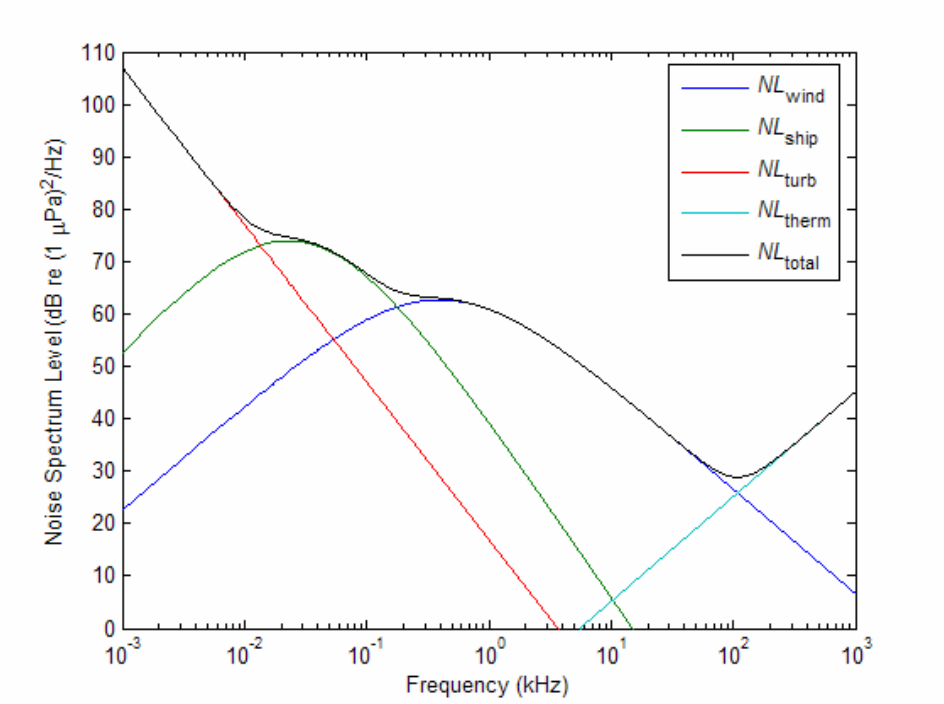

Figure 6 The noise spectrum level (*NSL*) in dB based on empirical formulas by Coates [5].

Figure 7 demonstrates that the wind speed has a profound effect on the value of the *NSL*.

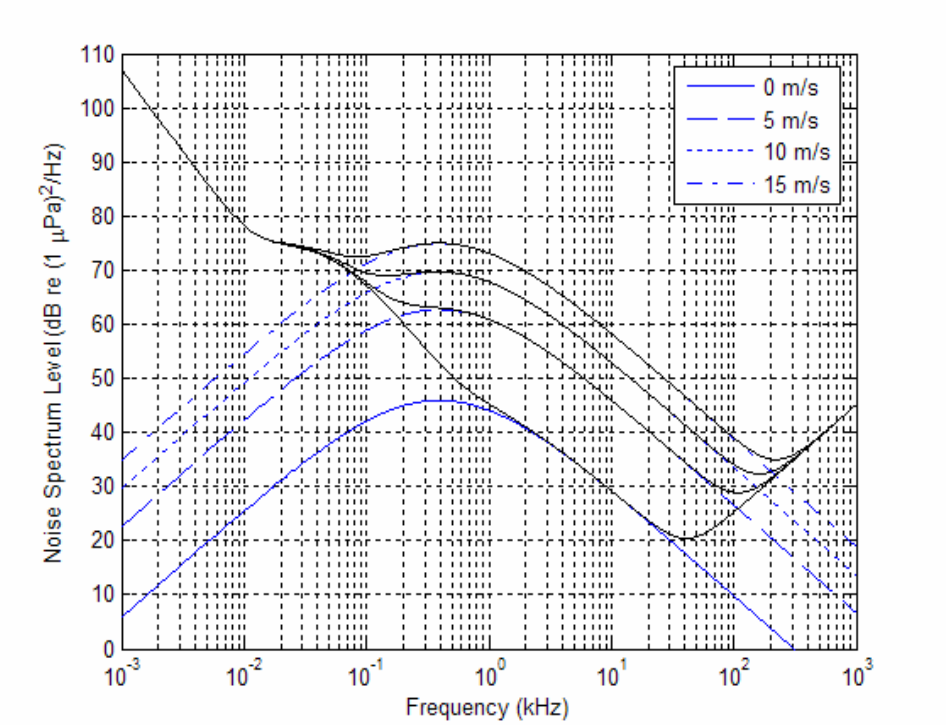

Figure 7 Increase of wind speed has a profound effect on the noise spectrum level. Based on empirical formulas by Coates [5].

3. Optimum Carrier Frequency as a Function of Range

Combining *TL* and *NL* for a wind speed of 5 meters per second (m/s) results in Figure 8, where $-(TL+NL)$ is plotted so that the red-yellow region indicates favorable transmission conditions. For the maximum anticipated range of 500 m, small negative values for $-(TL+NL)$ can be observed somewhere between 30–50 kHz. In order to determine an exact minimum, a slice at 500 m is taken from Figure 8 and plotted for various wind speeds and medium shipping density, resulting in Figure 9.

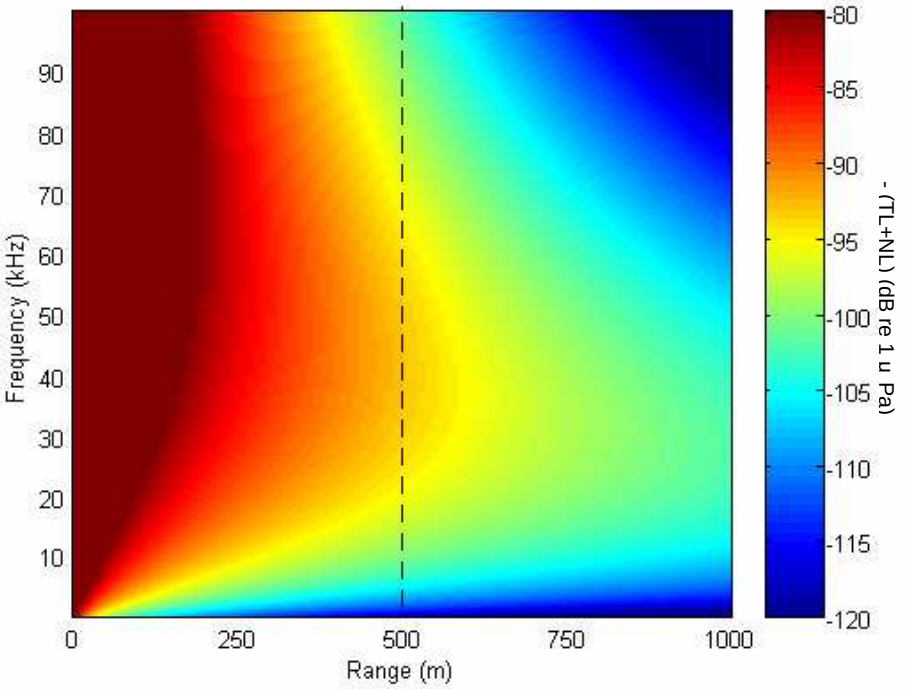

Figure 8 Effect of frequency and range on transmission loss and noise level (dB re 1 μPa).

Table 1 shows the frequencies at which the minimum TL+NL are found for various wind speeds and seawater temperatures with medium shipping density, a salinity of 35 ppt and a water depth of 50 m. The last column shows the frequencies for various wind speeds when averaged over sea water temperature. These averages are taken to be representative of "typical" values. It is obvious from Table 1 that the minimum value for $TL + NL$ depends for a large amount on wind speed and to a lesser extent on sea water temperature. Other factors, like shipping density or water depth, have no significant influence.

	8˚C	**14**˚C	**20**˚C	**26**˚C	**Average**
0 m/s	28.3	29.0	30.1	31.4	29.7
5 m/s	39.3	38.9	41.1	44.3	40.9
10 m/s	40.3	39.6	41.8	45.3	41.8
15 m/s	40.5	39.7	42.0	45.6	42.0

Table 1 Frequency (kHz) at which $TL + NL$ minima occur for various wind speeds in m/s (rows) and sea water temperatures in degrees Celsius (columns).

In the case of no wind, the frequency at which $TL + NL$ has a minimum value is approximately 30 kHz. In the presence of wind speeds of 5–15 m/s, this frequency can be found just above 40 kHz. We will define the frequency at which the least losses occur for a given range, the optimum carrier frequency. In order to ensure reliable communications, it is desirable to choose this frequency as the carrier frequency or center frequency when considering bandwidth.

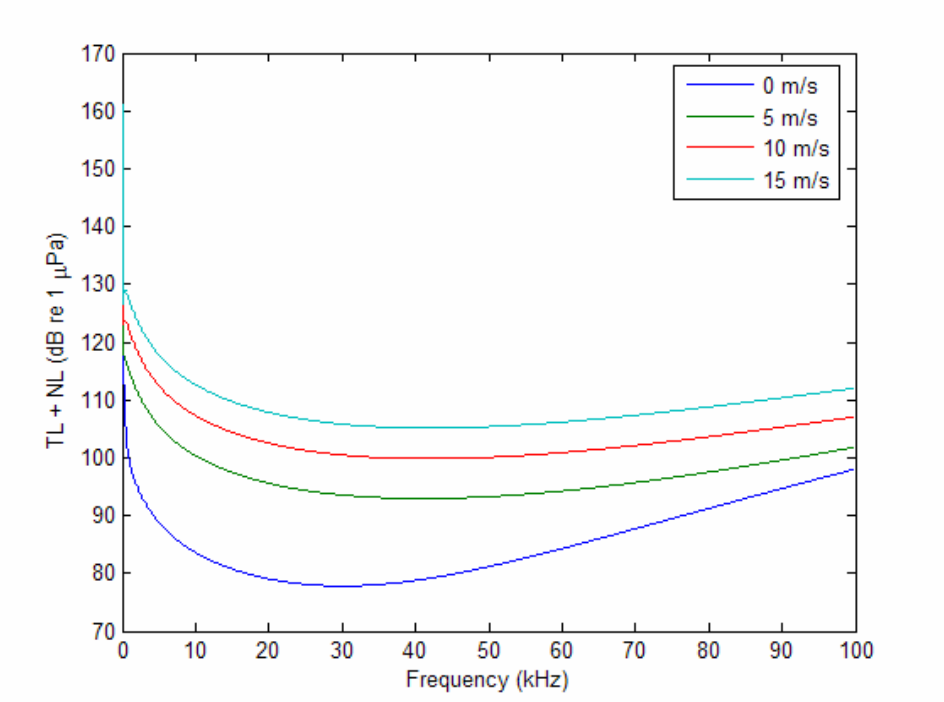

Figure 9 $TL + NL$ in dB versus frequency in kHz for various wind speeds at a range between source and receiver of 500 m and sea water temperature of 20˚C, $S = 35$ ppt, $pH = 8.0$ and $D = 50$ m. The optimum carrier frequency (kHz) is found at the minimum value of $TL + NL$.

4. Source Level

Now that *TL*, *NL* and optimum carrier frequency are known, the *SL* required to produce a desired SNR_a in dB can be determined. Although high source levels (>180 dB) are possible and common in underwater acoustic communications, they introduce reverberation which prolongs the impulse response. Reverberation cause symbols to

overlap at the receiver, which is called intersymbol interference. Avoiding intersymbol interference by reducing reverberation can be achieved by decreasing the *SL*. Other motivations for minimizing *SL* are transmission security and energy budget. It is therefore relevant to balance the choice of transmit power with the expected *TL* and *NL* and desired SNR_a.

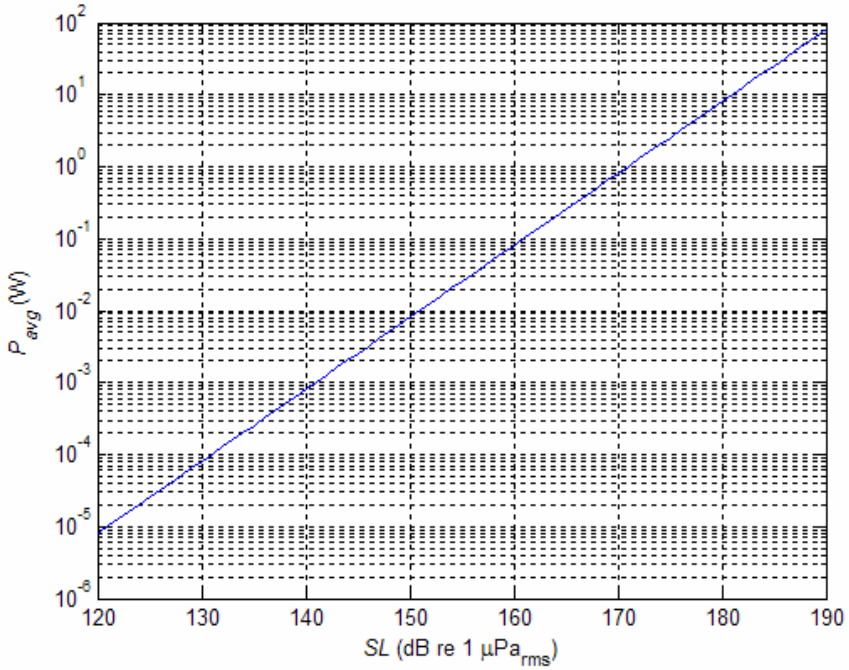

Figure 10 Time-average acoustic power P_{avg} in W versus *SL* in dB for P_{ref} = 1µPa, R_{ref} = 1 m for ρ = 1026kg/m^3 and c = 1499m/s.

The time-averaged radiated acoustic power in watts for a time-harmonic, omnidirectional, point source related to this SL is given by

$$P_{avg} = \frac{4\pi R_{ref}^2 \cdot P_{ref}^2}{\rho c} \cdot 10^{SL/10} \tag{2.18}$$

where P_{ref} =1 µPa is the root mean square reference pressure, R_{ref} = 1 m is the reference range from the source, ρ is the density of the medium in kg/m^3 and c is the speed of sound in m/s in that medium at a given depth, salinity and temperature (see Figure 10).

III. PHYSICAL LAYER

The Open Systems Interconnection (OSI) reference model, which was developed for terrestrial purposes by the International Organization for Standardization (ISO) consists of seven layers of standards that can be developed independently and simultaneously. Data moves down through these layers before being transmitted and moves up again at the receiver. Although this model is rarely achieved in practice, its structure has proven useful for the development of network protocols. We will use this model as a way to guide us through the process of studying Seastar networking aspects by focusing only on the lowest three layers: the physical layer, link layer and network layer.

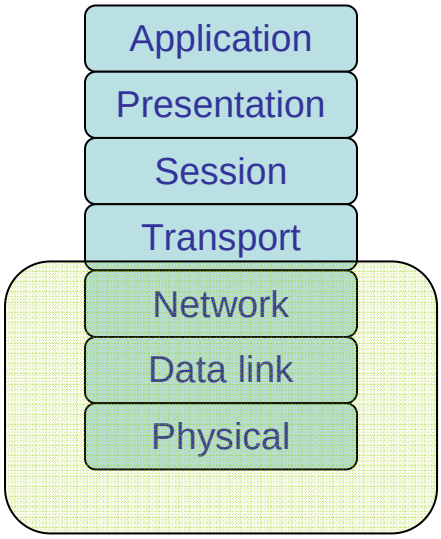

Figure 11 OSI model

The physical layer describes the transmission of digital symbols over the physical medium and deals with mechanical, functional, structural and procedural characteristics to access the medium. The challenge within the physical layer is to use the very limited bandwidth that is available in the underwater channel as efficiently as possible.

A. MODULATION

An issue that needs to be addressed is finding a suitable waveform modulation type. Stojanovic [18] and Akyildiz, et al [19] provide useful summaries of research and developments in the field of underwater acoustic communications. Intersymbol interference due to multi-path arrivals and a time-varying channel seriously degrade communication performance. Since mobile nodes are not excluded from being part of a Seastar network, the Doppler effect due to motion of transmitter and/or receiver is another factor that contributes significantly to performance degradation. Frequency shift keying (FSK) modulation techniques using phase-incoherent demodulation techniques are least sensitive to these channel fluctuations and are traditionally used for underwater communications. Recently, coherent demodulation techniques to detect phase shift keying (PSK) modulation and quadrature amplitude modulation (QAM) have demonstrated feasibility for use under water [18, 19]. Various forms of PSK and QAM using coherent demodulation have shown a bit rate increase of an order of magnitude compared to modulation types that depend on incoherent detection techniques [18, 19]. The experimental conditions, however, were mostly either very short range (<100m) in the vertical direction or very deep water with rather complex, often non-real-time detection, equalization and signal processing techniques at the receiver [18, 19]. Tests using existing commercial modems with PSK modulation conducted in the anechoic chamber and anechoic water tanks at NPS demonstrated a very poor performance, as will be discussed in Chapter IV, whereas MFSK modulation was generally reliable. Although the developments in the field of coherent demodulation look promising, it is so far considered not yet feasible for Seastar purposes. We will therefore focus on modulation techniques that do not depend on coherent detection and accept the bit-rate limitations.

1. M-ary Frequency Shift Keying (MFSK)

MFSK has proven to be a robust modulation scheme for underwater communications under various conditions [18, 19]. MFSK [20–23] uses multiple (*M*) frequencies, offset from the carrier frequency, to represent *M* different symbols, each

containing n_b bits so that $M = 2^{n_b}$. An MFSK signal is a pulse train as shown in Figure 12 (e.g., see [20]) and is represented in the time-domain by

$$x(t) = \sum_{n=1}^{N} x_n\left(t - t_n\right), \qquad 0 \le t \le T_d \tag{3.1}$$

where the n^{th} pulse, representing one of the M symbols, is given by

$$x_n\left(t\right) = A\cos\left(2\pi[\, f_c + \Delta f_n]t + \varepsilon_n\right)\text{rect}\left(\frac{t - 0.5T}{T}\right) \tag{3.2}$$

where

N: total number of pulses (symbols) transmitted in time interval $0 \le t \le T_d$

t_n: time instant in seconds when the n^{th} pulse begins

T_d: total duration in seconds of the transmitted signal (pulse train)

f_c: carrier frequency in hertz

Δf_n: frequency offset of n^{th} pulse in hertz representing a unique symbol

ε_n: unintentional additional phase shift of n^{th} pulse in radians

T: pulse length (symbol duration) in seconds of an individual pulse (symbol) in the pulse train. One symbol is equal to n_b bits.

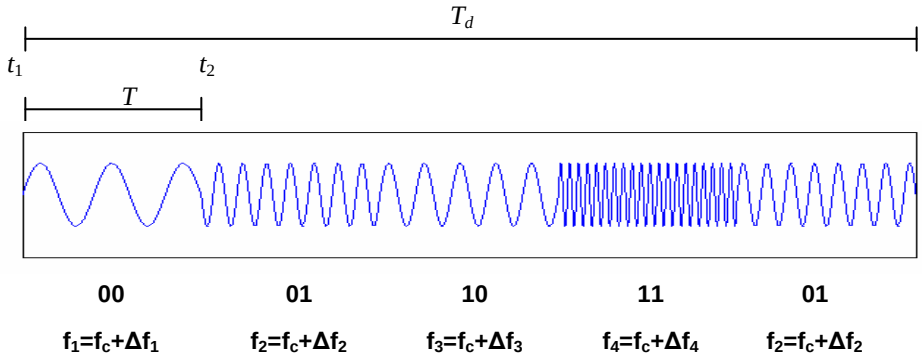

Figure 12 An example of 4-ary frequency shift keying (MFSK) using $M = 4$ frequencies to represent $M = 4$ different symbols.

In (3.2), a rectangle function is used to shape the pulse, but other windowing functions are not excluded. The duration of a symbol T is given by

$$T = n_b T_b , \tag{3.3}$$

where T_b is the bit duration in seconds. The total signal duration T_d is then expressed as NT seconds. The additional phase, ε_n, is included because unintentional phase shifts at the transmitter are hard to avoid. If the frequency offset is

$$\Delta f_n = \frac{k_n}{T}, \tag{3.4}$$

where k_n is a positive or negative integer, then the individual pulses are orthogonal even with phase shift ε_n, that is (e.g. see [21]),

$$\left\langle x_m(t), x_n(t) \right\rangle = \int_{-\infty}^{\infty} x_m(t)\, x_n^*(t)\, dt = E_{x_m} \delta_{m,n}, \qquad m,n = 1,2,...,M , \tag{3.5}$$

where $\delta_{m,n}$ is the Kronecker delta and E_{x_m} is the energy contained in symbol m given by

$$E_{x_m} = \left\langle x_m(t), x_m(t) \right\rangle = \int_{-\infty}^{\infty} \left| x_m(t) \right|^2 dt = \frac{1}{2} A^2 T, \qquad m = 1,2,...,M . \tag{3.6}$$

The factor k_n is an integer that determines the spreading of the frequencies. One possible set, the one that minimizes bandwidth and allows the bandwidth to be centered around an optimum carrier frequency, is $k_n \in \{\pm 1, \pm 2, \ldots, \pm M/2\}$ and is used for this analysis. Other sets such as $k_n \in \{\pm 1, \pm 3, \ldots, \pm (M\text{-}1)\}$, which spread the frequencies more sparsely, are also allowed.

Now that we know the representation of the signal in the time domain and the conditions for orthogonality, we will proceed by finding an analytical expression for an MFSK signal that relates its bandwidth to bit rate and the number of bits per symbol n_b. The first step is to take the Fourier transform of (3.1).This gives

$$\begin{aligned} X(f) = {} & \frac{AT}{2} e^{+j\pi(f+f_c)T} \sum_{n=1}^{N} \operatorname{sinc}\left\{ \left[f - \left(f_c + \Delta f_n \right) \right] T \right\} e^{+j\pi \Delta f_n T} e^{-j2\pi f n T} + \\ & \frac{AT}{2} e^{+j\pi(f-f_c)T} \sum_{n=1}^{N} \operatorname{sinc}\left\{ \left[f + \left(f_c + \Delta f_n \right) \right] T \right\} e^{-j\pi \Delta f_n T} e^{-j2\pi f n T} . \end{aligned} \tag{3.7}$$

The frequency spectrum is therefore a series of sinc-functions with maxima at $f = f_c + \Delta f_n$ and zero crossings that overlap each other exactly. In general, $\operatorname{sinc}(fT)$

equals zero at $f = i/T$ where $i = \pm 1, \pm 2, \ldots$, and exists for all frequencies. An infinite bandwidth is not realistic and therefore a judgment call is required for defining bandwidth. Note that bandwidth is always measured along the positive frequency axis. A rule of thumb for unit-amplitude sinc-functions is that the bandwidth is equal to a frequency interval where $|\text{sinc}(fT)| \geq 0.1$. As Figure 13 shows, $|\text{sinc}(fT)| < 0.1$ for $f > 3/T$, which refers to the location of the third zero crossing of the baseband sinc-function. Choosing the location of the next zero crossing, such as $f = 4/T$, or even $f = 5/T$, results in a more conservative estimate of signal bandwidth.

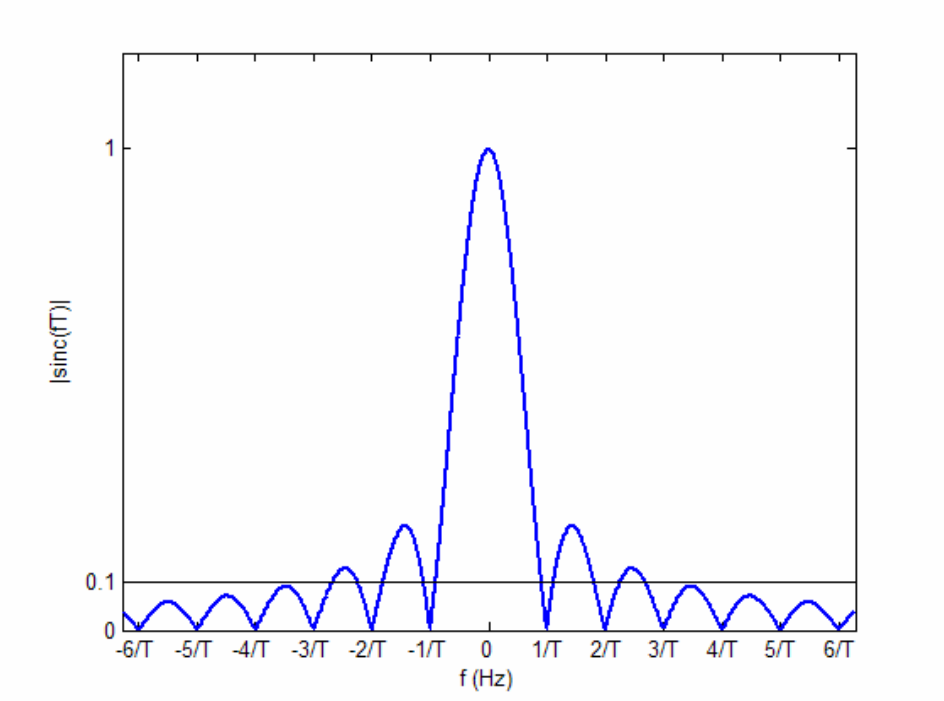

Figure 13 Magnitude spectrum of $\text{sinc}(fT)$.

Although it is often common to determine the bandwidth for a baseband sinc-function using $f = 1/T$, it is really an underestimation. In this thesis, we will choose a conservative estimate of bandwidth set at $f = 5/T$ for baseband, unit-amplitude, sinc-functions, although we do study the impact of other values later in this section.

The bandwidth of the MFSK frequency spectrum given by (3.7) is determined next. An estimate of the maximum positive frequency component f_{max} of (3.7) is

$$f_{\max} = f_c + \max \Delta f_n + \frac{5}{T}, \tag{3.8}$$

which, upon substitution of (3.4) and $k_n = M/2$ into (3.8), gives

$$f_{\max} = f_c + \frac{M}{2}\frac{1}{T} + \frac{5}{T} \tag{3.9}$$

An estimate of the minimum positive frequency component f_{min} of (3.7) is

$$f_{\min} = f_c + \min \Delta f_n - \frac{5}{T} = f_c - \frac{M}{2}\frac{1}{T} - \frac{5}{T} \tag{3.10}$$

Therefore, an estimate of the required bandwidth for transmitting an MFSK signal is

$$BW_x = f_{\max} - f_{\min} = \frac{M}{T} + \frac{10}{T} = (M+10)\frac{1}{T} \tag{3.11}$$

Finally, upon introducing

$$D = \frac{1}{T}, \tag{3.12}$$

where D is defined as the symbol rate or baud in symbols per second, the bandwidth formula given by (3.11) can be rewritten as

$$BW_x = (M+10)D. \tag{3.13}$$

Substitution of (3.3) into (3.12) gives

$$D = \frac{R_b}{n_b} \tag{3.14}$$

where

$$R_b = \frac{1}{T_b}, \tag{3.15}$$

is the bit rate in bits per second (bits/s).

Equation (3.13) is an estimate of the bandwidth of an MFSK signal in terms of the symbol rate D and the number of unique symbols M. The definitions for bit rate and symbol rate or baud in the literature are sometimes confusing since it may be unclear whether these rates refer purely to information bits or to raw bits including headers and bits for coding and error detection/correction, generally summarized as overhead. The

definitions for D and R_b as stated in this thesis refer to symbols and raw bits including overhead, meaning that the actual information rate may very well be lower.

Before any conclusions are made on a suitable bandwidth for a Seastar network, (3.13) will be analyzed in more detail. For this analysis, $M = 2^{n_b}$ represents the number of different symbols that can be created based on the number of bits n_b. For example, if $n_b = 2$, then there are $M = 4$ possible symbols consisting of two bits: 00, 01, 10 and 11. This requires four offset frequencies. Note that both M and D depend on n_b, the number of bits per symbol. Therefore, the relation between bandwidth BW_x, symbol rate D, and M can also be expressed in terms of n_b and the bit rate R_b, and is shown in Figure 14. Rewriting (3.13) in terms of n_b gives

$$BW_x = \left(2^{n_b} + 10\right)\frac{R_b}{n_b}. \tag{3.16}$$

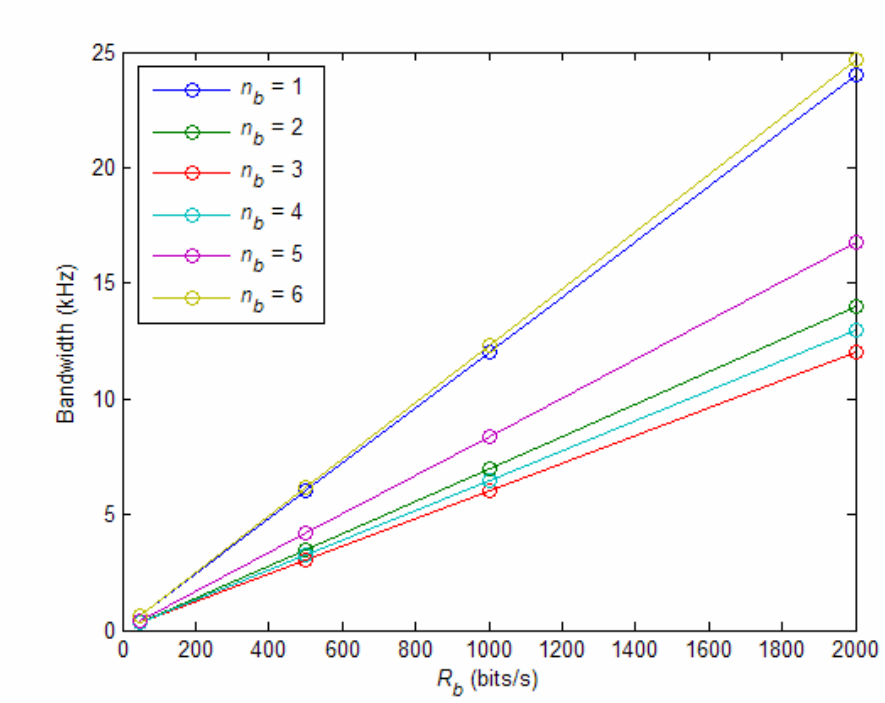

Figure 14 Bandwidth BW_x in kHz of a MFSK signal versus bit rate R_b in bits/s for different number of bits-per-symbol (n_b). A reduction of the number of frequencies Δf_n by reducing n_b does not always result in a smaller BW_x.

At first glance, Figure 14 shows what is expected: for a given value of n_b, increasing the bit rate comes at the cost of increasing bandwidth. Examining the figure

more carefully reveals an interesting phenomenon. As n_b increases, the number of frequency offsets Δf_n increases and one would expect an increase in signal bandwidth. This expected trend apparently breaks down for a fixed value of R_b and for small n_b and shows the opposite effect. The following example illustrates this numerically.

For $R_b = 1000$ bits/s and $n_b = 3$ bits/symbol, the transmission bandwidth is $BW_x = 6$ kHz. For $n_b = 5$ bits/symbol, $BW_x = 8.4$ kHz shows the expected increase in transmission bandwidth BW_x. If, on the other hand, a lower number of bits per symbol is chosen, for example $n_b = 2$ bits/symbol, we find $BW_x = 7$ kHz, which is larger than the bandwidth for $n_b = 3$ bits/symbol. This implies the existence of an optimum value for n_b for a given value of R_b.

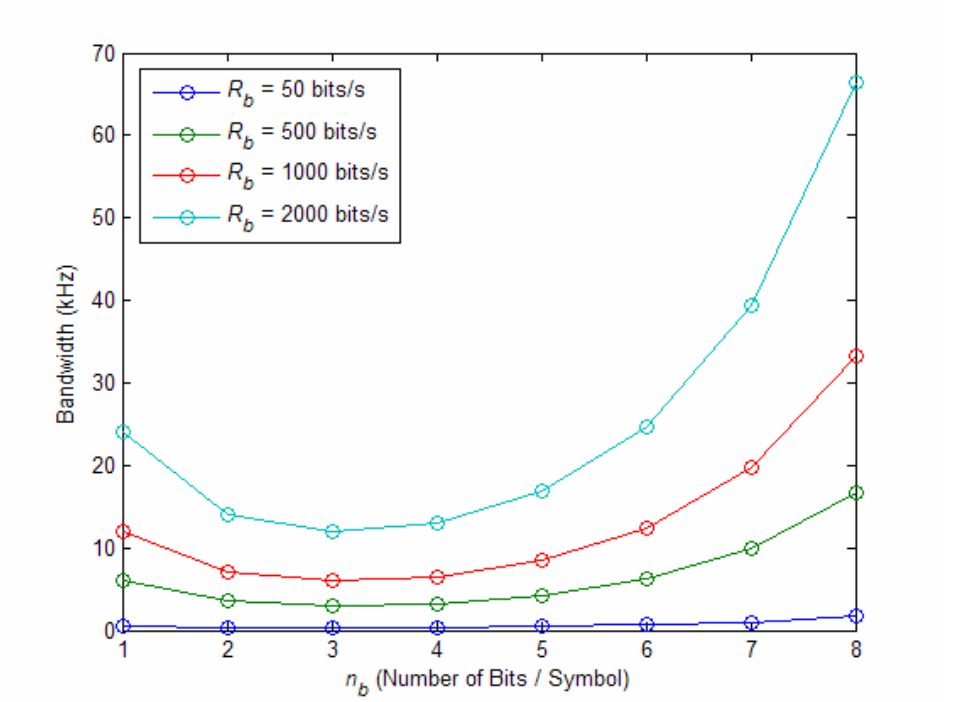

Figure 15 Bandwidth BW_x in kHz of a MFSK signal versus n_b, the number of bits per symbol for different bit rates R_b in bits/s. If a minimum transmission bandwidth is desirable, an optimum value for n_b can be found near $n_b = 3$.

The optimum value for n_b becomes obvious when plotting bandwidth BW_x versus n_b for a given R_b (see Figure 15). If (3.16) is rewritten as follows

$$BW_x = \left(2^{n_b} + 2z\right)\frac{R_b}{n_b}, \tag{3.17}$$

where z defines the number of zero crossings of the sinc-function (previously $z = 5$), then in order to find the optimum value of n_b that will minimize BW_x for a given value of R_b and z, we need to solve the following equation:

$$\frac{d}{d\,n_b} BW_x = \frac{R_b 2^{n_b}}{n_b}\left(\ln 2 - \frac{1}{n_b} - \frac{2z}{n_b 2^{n_b}}\right) = 0 \tag{3.18}$$

or

$$n_b \ln 2 = 1 + \frac{2z}{2^{n_b}}. \tag{3.19}$$

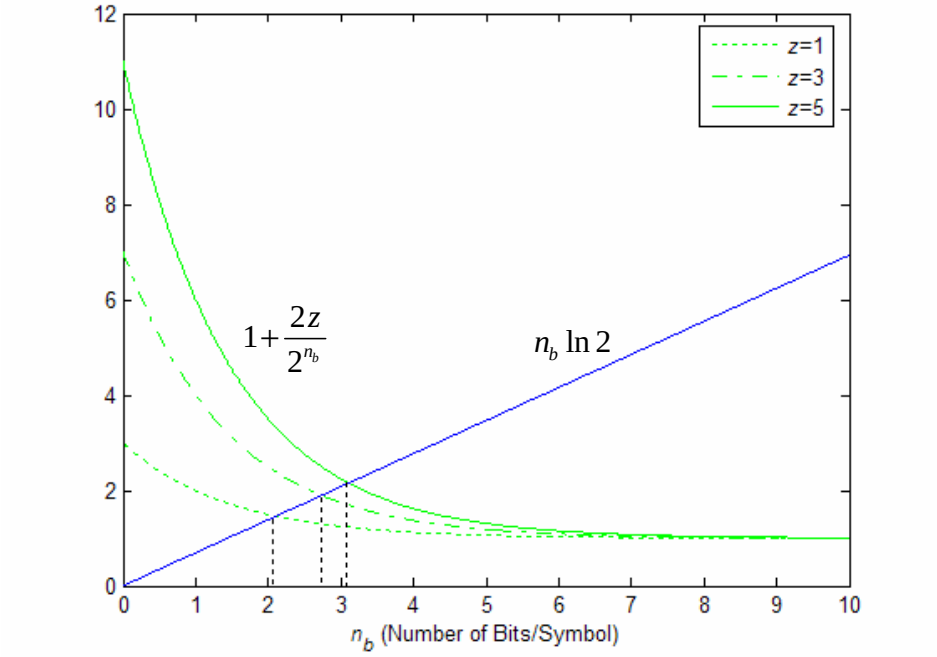

Figure 16 Graphical solution of the transcendental equation given by (3.19). A more conservative definition of bandwidth shifts the optimum value for n_b from 2 ($z = 1$) to 3 ($z = 3$ and $z = 5$).

Equation (3.19) is a transcendental equation which is independent of R_b. Figure 16 is a graphical solution of (3.19) and shows that the optimum value of n_b depends on the

choice for z and, therefore, on the definition of bandwidth. Rounding the values to the nearest integer shows that the optimum values for n_b are: $n_b = 2$ for $z = 1$, and $n_b = 3$ for $z = 3$ and $z = 5$. The existence of an optimum number of bits per symbol can be explained by the relatively small impact that the term 2^{n_b} in (3.16) has at low values of n_b compared to the $1/n_b$ factor. The term 2^{n_b} quickly starts to dominate at values of n_b greater than three.

The general impact of z on BW_x is shown in Figure 17. The definition of bandwidth, again, is subjective and allows for both over- and underestimation of achievable bit rates. The conservative choice of $z = 5$ results in a BW_x that is almost twice as large as a bandwidth definition based on the first zero crossing ($z = 1$).

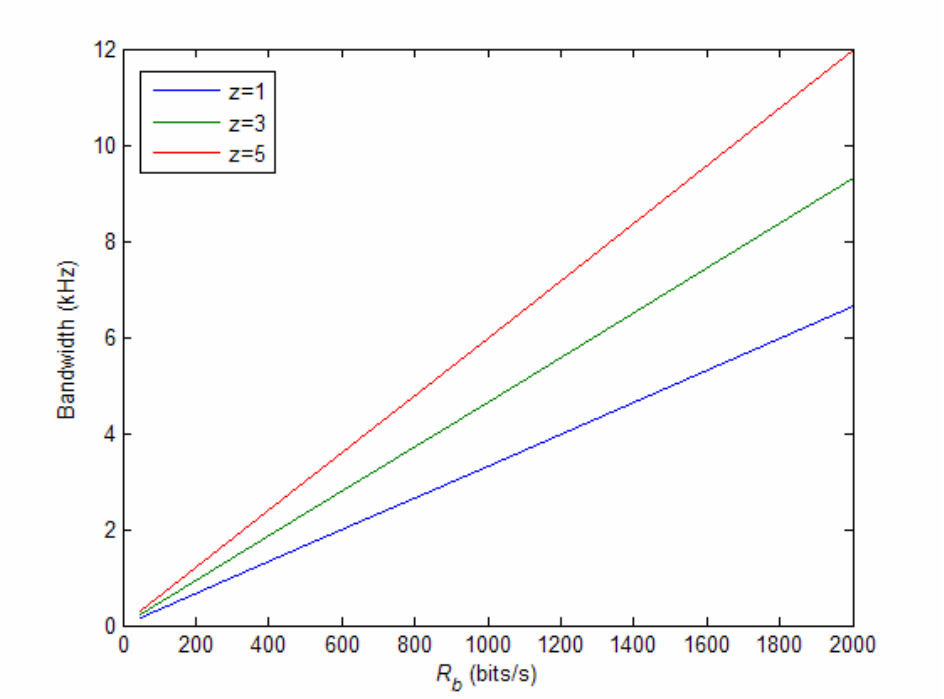

Figure 17 Influence of the choice for the number of zero crossings z on bandwidth BW_x for $n_b = 3$. The relation between BW_x in kHz, R_b in bits/s and z is given by (3.17).

We summarize this section by stating that a parametric analysis can be performed on MFSK signals. The analysis of (3.16) has revealed some interesting and useful relations between bandwidth BW_x, bit rate R_b, the number of symbols M, the number of

bits-per-symbol n_b and the definition of bandwidth based on (3.7). One should be careful and precise in defining the bandwidth for a sinc-function. This choice has a profound effect on the maximum transmission bandwidth and may even result in adjusting n_b when minimizing bandwidth. Equation (3.19) and Figure 16 show that for a bandwidth defined below the third zero crossing ($z = 1$ or 2), the optimum value for n_b is $n_b = 2$. For a bandwidth defined by $z = 3, 4$ or 5, the optimum value for n_b is $n_b = 3$. Although this optimization for n_b is useful for minimizing transmission bandwidth, Figure 14 shows that higher bit rates require larger bandwidths for given a value for n_b.

2. Orthogonal Frequency Division Multiplexing (OFDM)

The OFDM technique has been claimed to be one of the most promising future communications technologies for achieving high data rate and large system capacity [24] and is expected to be a valid and robust communications technique for underwater use [19]. Although the performance has only been tested marginally in an experimental setup [25], simulations [25, 26] indicate that the technique allows avoidance of intersymbol interference.

OFDM uses a multi-carrier modulation scheme to transmit broadband data in parallel over *N* orthogonal carriers which allows spectral efficiency and eliminates the use of guard bands between the carriers [24]. In this section, we will derive a formula for OFDM that allows us to analyze the relation between bandwidth versus bit rate and other characteristics. This will then be compared to the MFSK analysis that was done in the previous section.

The time domain equation for OFDM is given by Couch [27] as (see Figure 18)

$$x(t) = \sum_{n=0}^{N-1} x_n(t), \qquad 0 \le t \le T_d, \tag{3.20}$$

where

$$x_n(t) = A|w_n|\cos\left(2\pi[f_c + \Delta f_n]t + \angle w_n + \varepsilon_n\right)\text{rect}\left(\frac{t - 0.5T_d}{T_d}\right), \tag{3.21}$$

$$w_n = |w_n| e^{+j\angle w_n}, \tag{3.22}$$

$$\Delta f_n = \frac{n}{T_d}, \quad n = 0,1,...,N-1, \tag{3.23}$$

$$\text{rect}\left(\frac{t-0.5T_d}{T_d}\right) = \begin{cases} 1, & 0 \le t \le T_d \\ 0, & \text{otherwise,} \end{cases} \tag{3.24}$$

and where

N: total number of pulses (symbols) transmitted in time interval $0 \le t \le T_d$

w_n: complex-valued input symbol

T_d: total duration in seconds of the transmitted OFDM signal

f_c: carrier frequency in hertz

Δf_n: frequency offset in hertz

ε_n: possible, unwanted phase shift in radians.

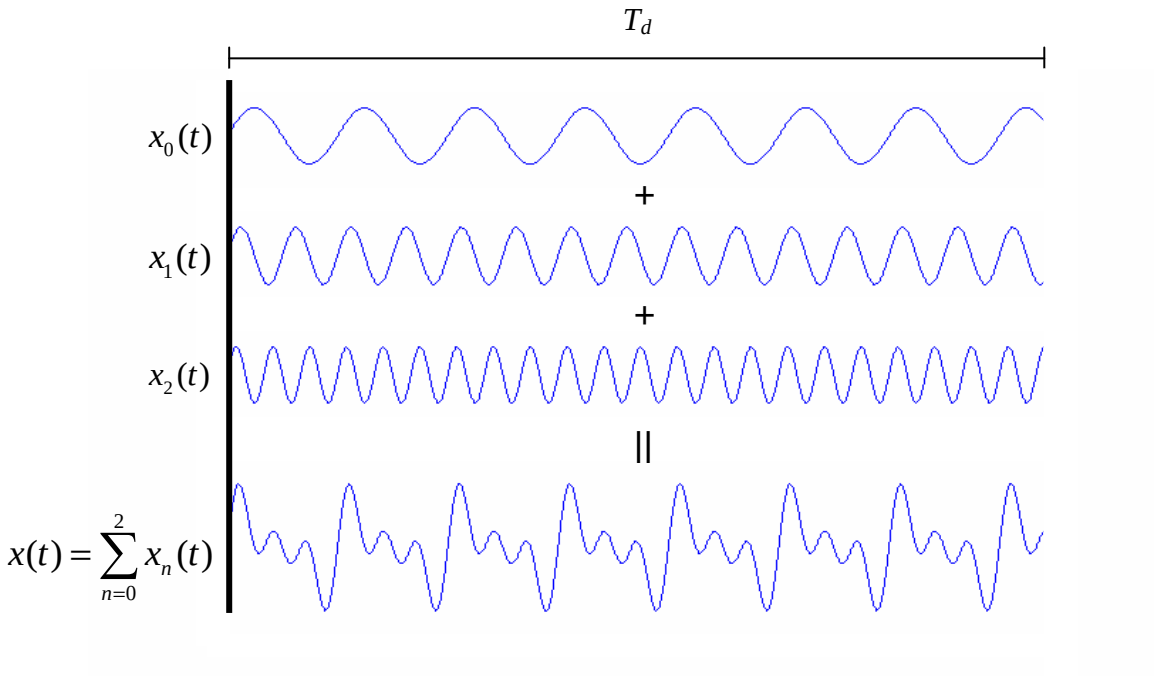

Figure 18 An example of OFDM with $N = 3$ sub-carriers transmitting for T_d seconds.

The relation between the total duration T_d of N simultaneous transmitted pulses, the duration of an input symbol T in seconds and the bit duration T_b in seconds is

$$T_d = NT \tag{3.25}$$

where

$$T = n_b T_b \tag{3.26}$$

and where n_b is the number of bits per symbol. The offset frequencies do not refer to unique symbols as with MFSK. Instead, symbol modulation is achieved by varying the amplitude ($|w_n|$) and/or the phase ($\angle w_n$) of a sub-carrier.

Taking the Fourier transform of (3.20) results in the following expression for the complex frequency spectrum of an OFDM transmitted signal:

$$\begin{aligned} X(f) = &\frac{AT_d}{2} e^{-j\pi(f-f_c)T_d} \sum_{n=0}^{N-1} |w_n| \operatorname{sinc}\left\{\left[f-(f_c+\Delta f_n)\right]T_d\right\} e^{+j\pi\Delta f_n T_d} e^{+j\angle c_n} + \\ &\frac{AT_d}{2} e^{-j\pi(f+f_c)T_d} \sum_{n=0}^{N-1} |w_n| \operatorname{sinc}\left\{\left[f+(f_c+\Delta f_n)\right]T_d\right\} e^{-j\pi\Delta f_n T_d} e^{-j\angle c_n} \end{aligned} \tag{3.27}$$

where

$$\angle c_n = \angle w_n + \varepsilon_n \,. \tag{3.28}$$

As with MFSK, this frequency spectrum is a series of sinc-functions with maxima at $f = f_c + \Delta f_n$ and zero crossings that overlap each other exactly. Recall that $\operatorname{sinc}(f T_d) = 0$ at $f = i/T_d$ where $i = \pm 1, \pm 2, \ldots$, and that $|\operatorname{sinc}(f T_d)| < 0.1$ for $f > 3/T_d$ (see Figure 13) which refers to the location of the third zero crossing of the baseband sinc-function. In contrast to MFSK, f_c is not the center frequency – it is the lowest frequency $(n = 0)$, thus making the maximum frequency offset

$$\max \Delta f_n = \frac{N-1}{T_d} \tag{3.29}$$

and the minimum frequency offset

$$\min \Delta f_n = 0 \,. \tag{3.30}$$

The bandwidth is defined as

$$BW_x = f_{\max} - f_{\min} = \left(f_c + \max \Delta f_n + \frac{z}{T_d} \right) - \left(f_c + \min \Delta f_n - \frac{z}{T_d} \right) \tag{3.31}$$

where z is the integer number of zero crossings of the sinc-function that are used to estimate both the maximum and minimum frequency components. Substitution of (3.29) and (3.30) into (3.31) gives

$$BW_x = \frac{N + 2z - 1}{T_d}, \tag{3.32}$$

which, upon substituting (3.25), can be rewritten as

$$BW_x = \left(1 + \frac{2z - 1}{N} \right) \frac{1}{T}. \tag{3.33}$$

The input symbol rate, or baud, is (see [27])

$$D_{in} = \frac{1}{T} = \frac{1}{n_b T_b} = \frac{R_b}{n_b}, \tag{3.34}$$

where R_b is the bit rate in bits/s. Finally, when (3.34) is substituted into (3.33), the following analytical expression for an OFDM transmitted signal relating its bandwidth to bit rate is found:

$$BW_x = \left(1 + \frac{2z - 1}{N} \right) \frac{R_b}{n_b}. \tag{3.35}$$

The next step is to make a fair comparison with MFSK. One way to do this is to demand the same number of symbols N to be transmitted over a certain time period T_d for both transmission schemes and compare BW_x. It can be seen from (3.1) and (3.20) that MFSK transmits N symbols in series in the time interval T_d as short pulses, each with duration T, whereas OFDM transmits the same number of symbols in parallel with duration T_d. Having thus ensured that a fair comparison is possible, we start the OFDM parametric analysis by setting N as the variable and observing the relation with BW_x. This is done simultaneously for $z = 1, 3$ and 5, $n_b = 3$, and $R_b = 2000$ bits/s as shown in Figure 19. Two important trends can be observed from Figure 19.

First, it shows that increasing the number of sub-carriers reduces the bandwidth. In other words, transmitting long symbol sequences (large *N*) in parallel is favorable in terms of bandwidth reduction. Note from (3.1) and (3.25) that this increases the total transmission duration T_d for both modulation schemes in the same amount.

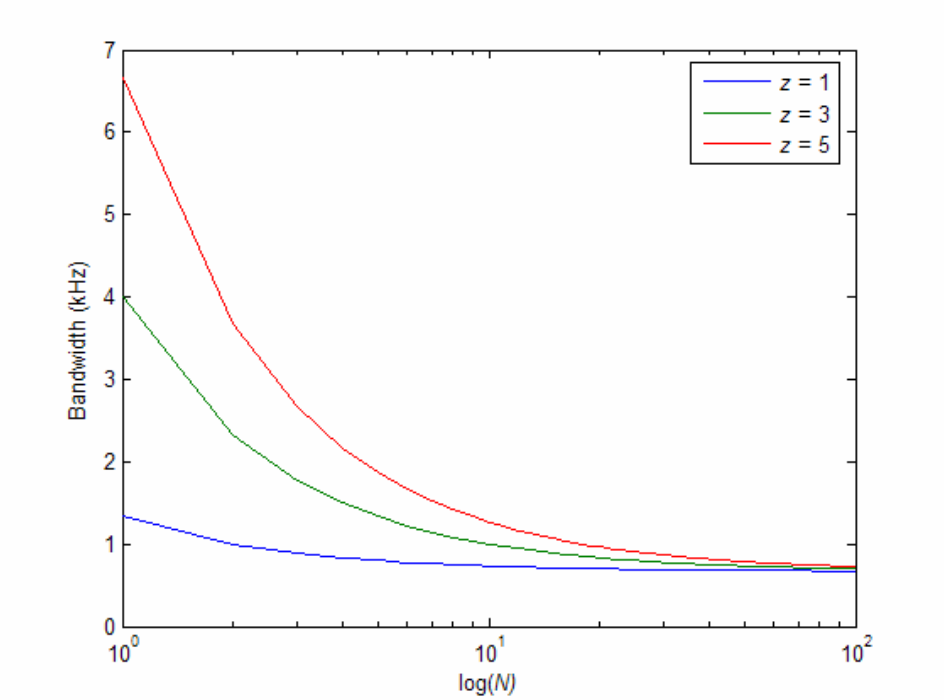

Figure 19 Bandwidth BW_x in kHz versus number of OFDM sub-carriers *N* for $R_b = 2000$ bits/s, $n_b = 3$, and $z = 1, 3$ and 5.

The second trend that is observed from Figure 19 is that the bandwidth for any *z* reduces asymptotically to the same value, which for $R_b = 2000$ bits/s and $n_b = 3$ is $BW_x \approx 667$ Hz. Recall that $z = 1$ is most commonly used, $z = 3$ represents the case where $|\text{sinc}(f T_d)| < 0.1$, and $z = 5$ is the conservative case. This asymptotic behavior is also described by Couch [27] and by using (3.35) and (3.34) translates in our case as follows:

For $z = 1$:

$$BW_x = \left(1 + \frac{1}{N}\right) D_{in} \quad \text{and if } N > 10, \text{ then } BW_x \approx D_{in}. \tag{3.36}$$

For $z = 3$:

$$BW_x = \left(1 + \frac{5}{N}\right) D_{in} \quad \text{and if } N > 50, \text{ then } BW_x \approx D_{in} \tag{3.37}$$

For $z = 5$:

$$BW_x = \left(1 + \frac{9}{N}\right) D_{in} \quad \text{and if } N > 90, \text{ then } BW_x \approx D_{in} \tag{3.38}$$

As a rule of thumb, it can thus be stated that if $N \gg (2z - 1)$, $BW_x \approx D_{in}$.

Keeping $n_b = 3$ and letting $N = 100$, the relation between BW_x and R_b for $z = 1, 3$ and 5 given by (3.35) is shown in Figure 20. Figure 20 shows that much higher bit rates can be obtained by using OFDM compared with MFSK (see Figure 17) for similar values of BW_x. It also confirms that $BW_x \approx D_{in}$ for large N and it shows that the choice for z does not have a similar impact on required BW_x as is the case with MFSK.

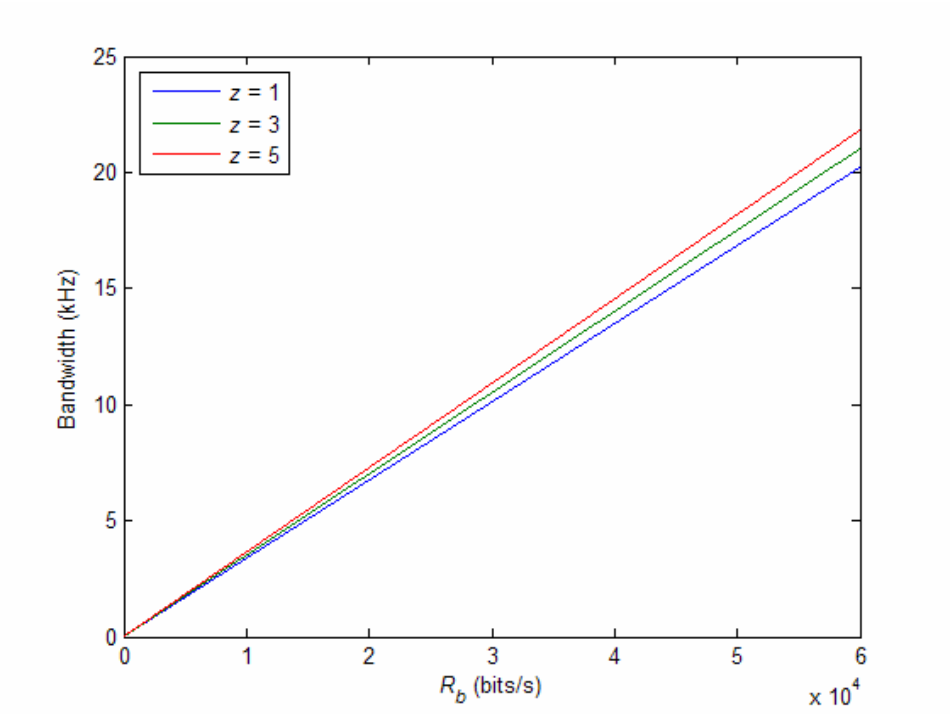

Figure 20 Bandwidth BW_x in kHz versus bit rate R_b in bits/s for OFDM with $n_b = 3$, $N = 100$, and $z = 1, 3$ and 5.

It may already be obvious from (3.35) that a large number of bits per symbol (n_b) reduces the bandwidth as is illustrated by Figure 21.

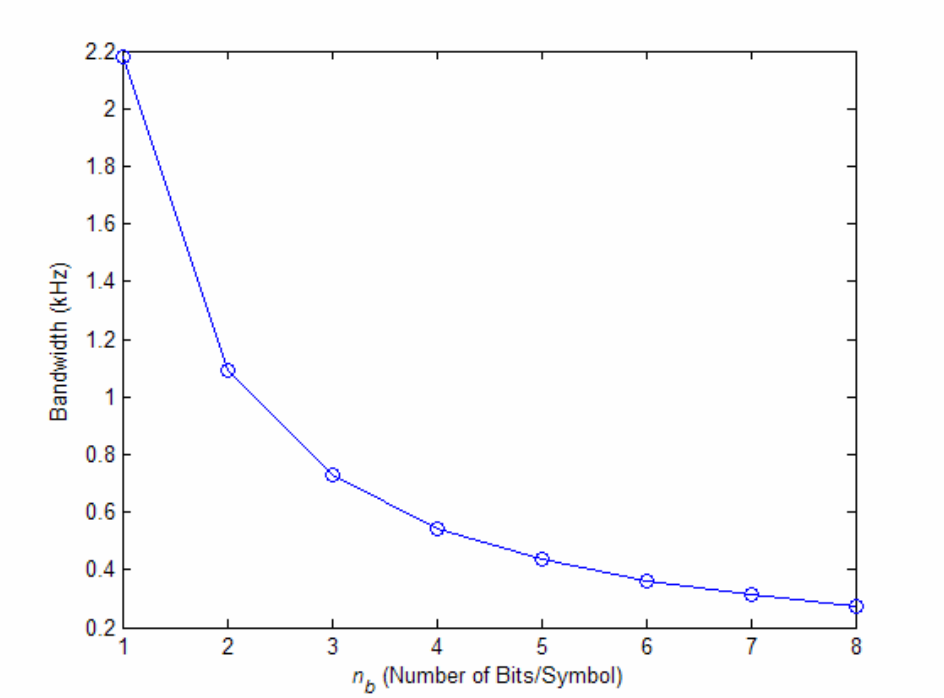

Figure 21 Bandwidth BW_x in kHz versus number of bits per symbol n_b for $N = 100$, $z = 5$ and $R_b = 2000$ bits/s.

Finally, Figure 22 compares OFDM to MFSK for $z = 1,\ 3$ and 5, $N = 100$, and $n_b = 3$. It can be seen that for a given R_b, the required OFDM bandwidth BW_x is always less than the required MFSK bandwidth and that the achievable OFDM bit rate is much higher than the MFSK bit rate (see Figure 17) for a given BW_x.

In summary, under similar conditions, OFDM requires significantly less bandwidth than MFSK to achieve a certain bit rate R_b. The biggest advantage of OFDM is that a high R_b for a given BW_x, or a small BW_x for a given R_b can easily be achieved by increasing the number of sub-carriers N. Although OFDM may seem the best modulation technique to use for Seastar, it must be emphasized that, to our knowledge, no in-water experiments have been conducted yet that confirm this behavior and that

demonstrate similar robustness in the underwater channel as MFSK. The developments of OFDM for underwater purposes need to be followed closely since OFDM has the potential to be a future candidate for Seastar applications.

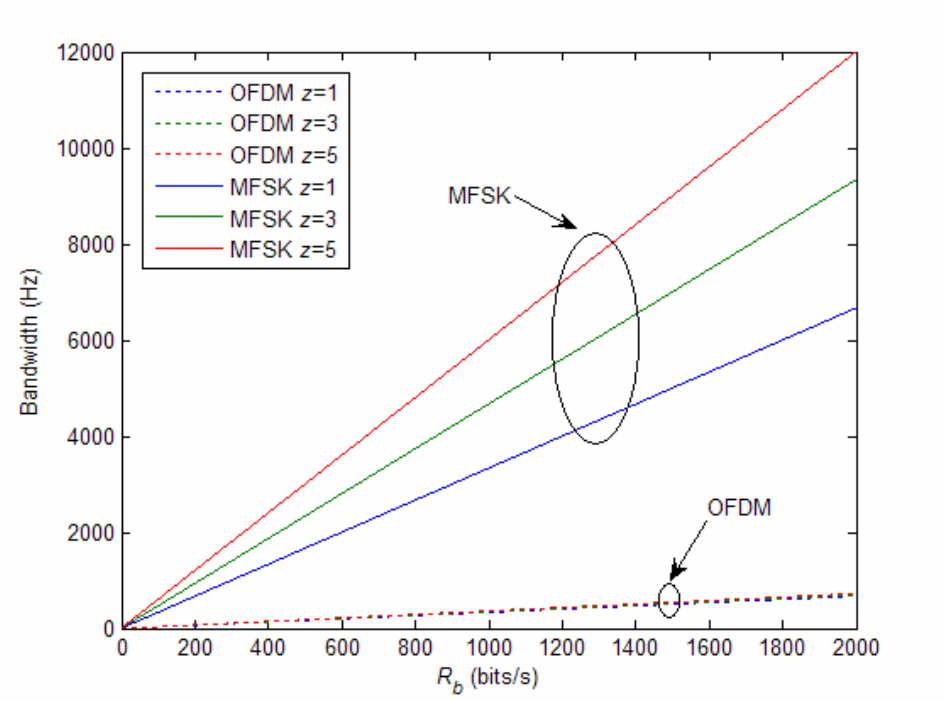

Figure 22 Bandwidth BW_x in Hz versus bit rate R_b in bits/s with $z = 1,\ 3$ and 5, $N = 100$, and $n_b = 3$ for OFDM and MFSK. To achieve a certain R_b, OFDM requires significantly less bandwidth than MFSK.

B. PRACTICAL CONSIDERATIONS

Seastar involves half-duplex communications, meaning that the individual nodes are not able to receive and transmit at the same time. Communication ranges within a Seastar network are confined to 500 m, by design. Mobile nodes like unmanned undersea vehicles (UUVs) or divers are not excluded from being part of such a network. Seastar anticipates an almost continuous information flow from an arbitrary number of peripheral nodes. A centralized configuration with a deterministic form of transmission control and a central, sophisticated, Seaweb access point, capable of fusing data and accepting data transmissions from cheap and unsophisticated nodes to reduce cost, introduces

asymmetry. Transmit and/or receive arrays containing multiple transducers are useful for steering or focusing a signal to avoid unwanted multiple arrivals and would provide array gain. Cost constraints, however, make the use of arrays—including the required signal processors at the peripheral nodes—highly unlikely. However, receive and/or transmit arrays at the sophisticated central node are not excluded.

1. Peer-to-Peer Communications

The centralized setup requires two-way communication between the central node and peripheral nodes. Depending on the preferred topology and potential need for node-to-node ranging and localization, communication amongst peripheral nodes may be an additional requirement. This imposes a further restriction on the geometry. Consider the case of a uniform radial distribution of peripheral nodes, as illustrated by Figure 23, for the special case of five nodes, where the central-to-peripheral node range in meters is given by

$$R = \sqrt{h^2 + (r/2)^2}\ , \tag{3.39}$$

where r is the peripheral node-to-node range in meters. In this geometry,

$$h = R\cos(\theta/2) \tag{3.40}$$

where θ is the angle that symmetrically distributes peripheral nodes around the central node as follows

$$\theta = \frac{2\pi}{n}, \tag{3.41}$$

with n being the number of nodes. At least six symmetrically spaced nodes are required to ensure communications at the maximum range between both central and peripheral as well as neighboring peripheral nodes. Calculating the range excess R', given by

$$R' = \frac{r-R}{R}\times 100\% = [2R\sin(\pi/n) - 1]\times 100\% \tag{3.42}$$

that is required in case of fewer nodes or the reduced range R_{red} to maintain neighboring node communications is trivial and results in Table 2. Using fewer nodes imposes no restriction when only central-peripheral communications are considered.

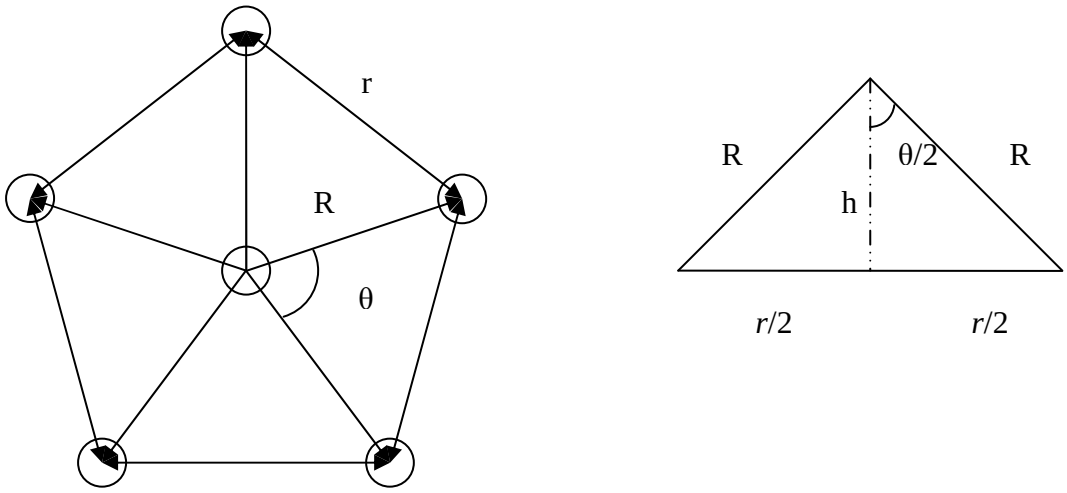

Figure 23 Network geometry with five symmetrically distributed nodes. At least six nodes are required to ensure $r \le R$.

n	R'	reduced R_{red} (r=500m)
2	100%	250m
3	73%	289m
4	41%	354m
5	18%	425m

Table 2 Required range excess and reduced range in case of less than 6 nodes.

2. Transmit Transducer

The next issue involves the practical realization of the transmission of a modulated waveform, mathematically represented by (3.1). A factor that severely limits the useful band is the transmitting sensitivity level or transmit voltage response (*TVR*) of a transducer. The *TVR* is defined as the ratio of the pressure response of a transducer to the applied voltage and is commonly expressed in units of dB re 1 μPa/V @ 1 m [7, 31]. In general the *TVR* of a transducer over its operating band is not constant. The usable bandwidth of a transducer is commonly described by the mechanical quality factor, which is defined as

$$Q_m = \frac{f_c}{f_u - f_l} \tag{3.43}$$

where f_c is the center frequency, or resonance frequency, and f_u and f_l are the upper and lower frequencies, respectively, at which the average power has dropped to one-half its value at the resonance frequency [7, 28]. A high Q_m represents a frequency response spectrum with a sharp peak, whereas a low Q_m represents a broader frequency response. The *TVR* will act as an additional filter on the transmitted waveform. As an example, a quick survey of commercially available transducers [29, 30] shows that a low Q_m of 2 to 3 is not unreasonable near the frequency band of interest.

Another limiting factor imposed by the transducer is the rise time for a pulse to reach steady state. Recall that (3.2) uses a rectangle function to describe a pulse. The rise time will affect the frequency spectrum that is transmitted, which has an impact on bandwidth and achievable data rates. It may also affect the orthogonality of the individual pulses depending on the modulation type. The rise time in seconds to reach 96% of the steady-state amplitude is give by [31]

$$t_{rise} = \frac{Q_m}{f_c}. \tag{3.44}$$

Another way of describing this is that Q_m cycles are required for the amplitude to build up to 96% of its final value, or reduce to 4% of its maximum value. Short pulses will be affected relatively more than long pulses.

3. Multi-access Interference

Although an individual LAN manages its own internal channel access, it needs to be considered that Seastar networks might operate in each other's vicinity. Interference has a dramatic impact on the performance, as will be shown in Chapter V, and frequency separation, resulting in bandwidth reduction, may be the only physical solution to overcome this problem.

C. PHYSICAL LAYER CASE STUDY

Let us, as an example, analyze the physical layer by performing a case study to identify a reasonable transmission bandwidth, capacity and energy budget. Doing so requires approximations and assumptions on several factors such as transducer performance and channel properties.

For wind conditions of 5–15 m/s shown in Table 1, the optimum carrier frequency can be found to be approximately 41 kHz. For convenience, we choose f_c=40 kHz. The practical bandwidth is governed by $TL + NL$ versus frequency spectrum and transducer properties. An additional consideration is that the frequency band should be separated from the Seaweb band (currently 9–14 kHz). Assuming that a Q_m of 2 is achievable, the half-power bandwidth based on (3.43) could stretch from 30–50 kHz which makes $BW_x = 20$ kHz. For a transducer with a Q_m of 2 and $f_c = 40$ kHz, applying (3.44) results in a rise time of 50 μs. If MFSK is the preferred modulation type, (3.13) can be applied to determine maximum achievable bit rate. If $n_n = 3$ bits/symbol, then $M = 2^3 = 8$, and the modulation type becomes 8-FSK. Based on (3.13) and (3.14), the maximum achievable symbol rate D is then approximately 1100 symbols/s which makes $R_b \approx 3300$ bits/s. Recall that overhead is included so the information rate will be lower by a factor that depends on the type of error detection and correction coding applied, message header lengths, etc. Using (3.12), the pulse length T for a symbol would have a minimum length of 0.9 milliseconds (ms) meaning that the ratio of t_{rise} over T would be 6%. If OFDM would be available, bit rates of $R_b \approx 55000$ bits/s are theoretically possible for the same $BW_x = 20$ kHz. Further analysis would be required to determine to what extent the signal would be affected by the rise time. It must be emphasized that the frequency spectrum of the transmitted signal given by (3.7) has been convolved by both the channel impulse response and the *TVR*.

The final step in our case study is to determine an energy budget for acoustic transmissions to a range of 500 m. To obtain a numerical value requires additional approximations and assumptions. Firstly, our analysis is limited to a single-frequency

time-harmonic signal only, and $\Delta f = 1$ Hz in (2.15). We also assume a specific SNR_a required at the receiver to reliably detect an incoming signal. Based on experience with commercially available Teledyne Benthos ATM-885 acoustic modems, a SNR_a of approximately 7 dB at the receiver input ensures a probability of detection of more than 95%. For a wind speed of 15 m/s and medium shipping density, a $TL + NL$ of 105 dB at $f_c = 40$ kHz can be extracted from Figure 9. However, the worst case $TL + NL$ for a single-frequency time-harmonic signal under these conditions within the 30–50 kHz band is 106 dB at 30 kHz. Adding the 7 dB SNR_a to the maximum $TL + NL$ value results in a required *SL* of 113 dB re 1 μPa @ 1 m [see (2.12)]. In order to find the input electrical power to achieve this *SL*, we need transducer *TVR* and impedance data, since the root-mean-square input electrical power, P_{rms}, in watts (W) [32] is expressed as

$$P_{rms} = i_{rms}^2 R = \frac{V_{rms}^2}{|Z|^2} R\,, \tag{3.45}$$

where i_{rms} and V_{rms} are the root-mean-square input current in amperes and voltage in volts, respectively. The variable *R* is the transducer resistance and |*Z*| is the magnitude of the transducer impedance in ohms. Although *Z* can be expressed in terms of resistance *R* and reactance *X*, it is more common to express the characteristics of a transducer in terms of admittance *Y* where

$$|Y| = \frac{1}{|Z|} = \sqrt{G^2 + B^2}\,, \tag{3.46}$$

with *G* and *B* being the conductance and susceptance, respectively. The units for *Y*, *G* and *B* are siemens (S). Equation (3.45) can now be rewritten as

$$P_{rms} = V_{rms}^2 G\,. \tag{3.47}$$

A limited survey of available transducers [29, 30] yields a *TVR* and *G* at the resonance frequency of 145 dB and 6500 μS, respectively, as reasonable values. The input root-mean-square power, required to produce a *SL* of 113 dB re 1μPa @ 1m can be found from

$$10\log(P_{rms}) = 20\log(V_{rms}) + 10\log(G) = SL - TVR + 10\log(G)\,. \tag{3.48}$$

Inserting the values for *SL*, *TVR* and *G* in (3.48) results in an input electric P_{rms} of 4.1 μW at 40 kHz provided that this is also the resonance frequency of the transducer to generate a *SL* of 113 dB. The energy required to transmit one symbol of duration $T = 0.9\,\text{ms}$, represented by a single frequency, is therefore $(4.1\ \mu\text{W}) \cdot (0.9\ \text{ms}) = 3.7 \times 10^{-9}\ \text{J}$.

Calculating the energy to receive a message is less trivial as it depends on the demodulator and amplifier used to detect and decode the signal. To keep our analysis general, we use a method for calculating energy consumption described in [33]. Although the energy efficiency of signal processors has increased over the years, it is likely that receiving signals at higher bit rates requires more energy. We therefore do not make assumptions on improvements of receiving power and take a typical value of 0.5 W based on [33] for the power required to receive data. In summary, to ensure transmission of a single frequency signal at 40 kHz over a range of 500 m at wind speeds of 15 m/s and medium shipping density with a commercially available transducer, requires an input electric root-mean-square power of 4.1 μW, further referred to as $P_{transmit}$. Reception of this signal under the same conditions requires 0.5 W, further referred to as $P_{receive}$.

The energy budget now fully depends on operational settings such as the type of network, the expected number of transmissions, bit rates, packet size and the number of modems that are part of this network. Nevertheless, it is useful to continue the analysis to ascertain the order of magnitude of the required energy budget. To finalize this analysis we consider a network consisting of one central modem receiving data packets from, and transmitting control packets to, six peripheral modems over fixed ranges of 500 m. The preferred modulation type is 8-FSK, each symbol consisting of 3 bits, and the available bandwidth is 20 kHz. Each peripheral modem is assumed to transmit one data packet of a fixed length of 2000 bytes with 8 bits per byte at 3300 bits/s once every cycle and the central modems transmit one 10-byte control packet at 500 bits/s to each peripheral modem each cycle. For each byte transmitted, an additional redundant byte is added and additional overhead created by headers, etc., is ignored. Each transmission is successful and no dead times between transmissions are considered. One transmission from a central modem to a peripheral modem therefore requires a period of

$$T_{control} = 2 \times 10 \text{ bytes} \times \frac{8 \text{ bits}}{\text{byte}} \times \frac{1\text{s}}{500 \text{ bits}} = 0.32 \text{ seconds}.$$

One transmission from a peripheral modem to the central modem requires a period of

$$T_{data} = 2 \times 2000 \text{ bytes} \times \frac{8 \text{ bits}}{\text{byte}} \times \frac{1\text{s}}{3300 \text{ bits}} = 9.70 \text{ seconds}.$$

The total required energy during one cycle for a peripheral modem is now

$$E_{total,peripheral} = T_{control} \cdot P_{receive} + T_{data} \cdot P_{transmit}$$
$$E_{total,peripheral} = 0.32\text{s} \cdot 0.5\text{W} + 9.70\text{s} \cdot 4.1\mu\text{W} = 0.16\text{J}.$$

The total required energy during one cycle for the central modem is now

$$E_{total,central} = \sum_{m=0}^{M} T_{data,m} \cdot P_{receive} + T_{control,m} \cdot P_{transmit}$$
$$E_{total,central} = 6 \cdot \left(9.70\text{s} \cdot 0.5\text{W} + 0.31\text{s} \cdot 4.1\mu\text{W}\right) = 29\text{J}.$$

The difference between required energy at the central modem and the energy required at any of the peripheral nodes is approximately two orders of magnitude. Although these values originate from many assumptions, the difference is significant enough to be taken into account. This difference, which is mainly due to the fact that receiving and processing a signal requires more power than transmitting, will further increase with a larger number of peripheral modems in the network. Added to this comes the fact that the central modem is also responsible for data fusion and transmission through a Seaweb network which makes the energy budget between central and peripheral modems even more asymmetric. From a peripheral modem's perspective, we have ignored the energy required by other components of the peripheral nodes, such as the sensor that provides data for transmission. Data transfer from sensor to modem and modulating this data requires additional energy that needs to be included in the peripheral modem's final energy budget.

An assumed battery capacity at the central modem of 300 watt-hours or 1.08 megajoules (MJ) that is fully available for the Seastar network would allow Seastar network operations for almost 26 days. This period would require a peripheral node battery capacity of 6000 joules (J) or 1.7 watt-hours permitting a smaller size and cost of these nodes.

The discussion regarding the physical layer of a Seastar network is summarized as follows. Seastar is able to operate at 500 m using a higher frequency spectrum than Seaweb for a period of about one month. The optimum carrier frequency is 40 kHz and, depending on the modulation type and signal processing techniques, bit rates of 3000 bits per second should be achievable using MFSK modulation and a spectral bandwidth of 20 kHz. The availability of OFDM could improve the performance by an order of magnitude. The energy budget for central and peripheral modems is asymmetric and consistent with the envisioned topologies. More accurate performance data requires a further study of the effects of the transducer and the communication channel on the waveform, and an update of energy consumption considering modems optimized for operation at the frequency band of interest.

IV. DATA LINK AND NETWORK LAYER

This chapter provides a framework for the organization of the dataflow within a Seastar network. The set of rules for data exchange is called a protocol and consists in our case of a layered structure as described in Figure 11. Extensive research in the field of access control during the last decade [19, 22, 23, 35–37] provides useful advice for designing Seastar. Network issues regarding error detection and correction that originate from terrestrial networks are considered for their applicability to Seastar. Finally, tradeoffs regarding topology, the physical arrangement of stations, are presented. Combining this information yields the design of a prototype Seastar network that is discussed in the last section.

A. DATA LINK LAYER

The link layer is responsible for ensuring transmission across the physical layer between two neighboring nodes and deals with synchronization, error control and flow control.

1. Access Control

Seastar is designed for multiple users to transmit information to a central node. In this multiple-access environment, it is necessary to share the transmission medium in a manner ensuring that packets are transmitted without interference from other network users. Research for applicability of terrestrial multiple access control techniques for underwater communication purposes [19, 22, 23, 35–37] provides useful tradeoffs. A distinction can be made in deterministic and random access methods.

Available deterministic access methods are frequency-division multiple access (FDMA), time-division multiple access (TDMA) and code-division multiple access (CDMA). FDMA [23] simply divides the available bandwidth into N sub-channels, where N depends on the number of nodes in the network. Since we anticipate a large number of nodes and a relatively small bandwidth, this access method is dismissed as

unsuitable. A theoretical variation of FDMA is spatial frequency reuse based on a cellular architecture [36]. This technique limits the user density and is therefore deemed impractical, but it could provide answers on questions regarding interference between clusters.

TDMA [23] provides the user with the full available bandwidth by allowing only one transmission at a time. The downside is that it creates an inefficiency because of the long time delays required in the underwater channel. Time dispersion of the signal further requires additional guard bands. Fixed time slots may decrease the efficiency even further when transmissions are shorter than the allocated time. Polling, or interrogating nodes by a master node, is a way to overcome this and avoid synchronization issues which keeps the complexity of nodes low. However, it introduces additional overhead which also decreases the efficiency.

CDMA [23] actually provides random access for users since it allows signal transmissions that overlap both in frequency as in time. It assigns a unique pseudo-random code sequence to each user by spreading the information signal across the entire frequency band. The receiver is able to demodulate the simultaneous transmitted signals because of the small cross correlation that the code sequences have with each other. Spreading can be achieved either by direct-sequence spread spectrum (DSSS) or frequency-hop spread spectrum (FHSS). FHSS requires less complex receivers and is more robust to multiple access interference than DSSS but is also more sensitive to Doppler shift effects. CDMA performance is sensitive to the relative receive power of simultaneous signals, and power control is required to mitigate this sensitivity. A recent CDMA experiment [37] has demonstrated that low complexity receiver algorithms are realizable and effective. Although CDMA appears to be a promising technique for underwater communications, especially in the case of a network with multiple users and moving nodes at short ranges such as Seastar, more research needs to be done in this field before it can be applied.

Random access methods like ALOHA, carrier sense multiple access (CSMA) and multiple access with collision avoidance (MACA) are reviewed in [19, 22, 23, 34]. Peripheral nodes using ALOHA and slotted ALOHA [22] generally do not “listen” to the

communications channel and transmit whenever data needs to be transmitted. Acknowledgement messages (ACK) report the reception of the message. CSMA [22] aims to prevent collisions by sensing communications activity and delaying new transmissions until the channel is clear. However, the long propagation time limits the effectiveness of CSMA for acoustic communications. MACA [22] uses request-to-send (RTS) and clear-to-send (CTS) messages to establish communications before transmitting the data packets. In a positive acknowledgement MACA protocol, if no ACK message is received after the transmission is completed, the full packet will be retransmitted until reception is acknowledged. In a negative acknowledgement MACA protocol, the transmitter assumes success unless it receives a repeat request. Seaweb is an example of successful implementation of MACA under water. The RTS-CTS messages could further be used as probe signals for adaptive modulation or power control. Although MACA reduces the amount of retransmissions significantly, it introduces additional overhead prior to every data transmission.

Although random access methods are flexible, they are not very suitable under water when an almost continuous flow of information between nodes at short range is expected. The large amounts of collision avoidance overhead or retransmissions will cause large delays and make the network inefficient.

Based on the previous outline, we shall pursue the use of TDMA as the favorable access method for Seastar. In order to overcome difficulties regarding synchronization and predefined time slots, and to avoid the need for synchronized clocks, it is necessary to introduce some form of central control. This might be provided by a polling mechanism or a token to be passed from node to node. This issue will be addressed in the section that discusses the network layer. In the meantime, developments in the field of CDMA in the underwater environment need to be followed closely for application to Seastar.

2. CRC, FEQ and SRQ

Error control refers to mechanisms to detect and correct errors that occur in transmissions. One of the most common error detection mechanisms is the cyclic

redundancy check (CRC). CRC is described in detail by Stallings [22] and is based on the calculation of a code that is a function of the bits being transmitted. This code is appended to the information packet and introduces a small amount of overhead.

Two error correction mechanisms, forward error correction (FEC) and selective-reject automatic repeat request (SRQ), will be briefly discussed from a Seastar perspective.

One approach is to prevent retransmission by introducing redundant bits so that the receiver is capable of correcting any detected errors. This is the principle of FEC. While we do not want to divert towards a discussion regarding available coding techniques for FEC, it may be obvious that more redundancy comes at a cost. The term code rate which is related to FEC refers to a metric that expresses the overhead required to carry data at the same data rate as without the code. Code rates of 1/2, meaning that twice the bits are required, are no exception. FEC may take many forms and tradeoffs regarding overhead versus probability of error should be considered.

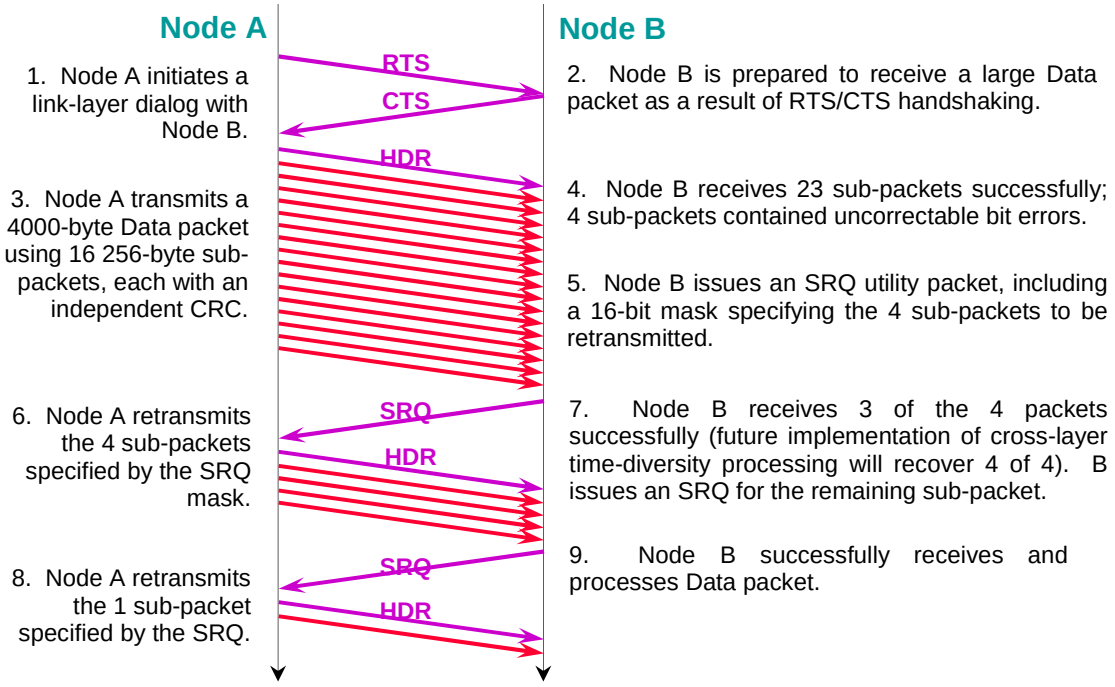

Figure 24 An example of selective automatic repeat request (SRQ) (After [38]). Retransmission of corrupted sub-packets continues until the full packet has been received successfully.

SRQ is a form of error control that is already successfully incorporated in Seaweb and is well documented by Kalscheuer [38]. SRQ relies on the detection of bit-errors by the receiver using CRC and results in retransmission of the corrupted data. The principle of selectivity refers to the possibility of retransmitting only a portion of the message instead of the full message. This requires that the data packet be divided into smaller sub-packets each padded with its own CRC bytes. The disadvantage of SRQ is that it incurs latency and overhead.

Both FEC and SRQ can be applied simultaneously as has been done for Seaweb. Harris et al. [3] studied the combined effects of applying FEC and dividing the packet into smaller sub-packets. Both FEC and SRQ have proven effective and are simultaneously suitable for Seastar applications. The effects of sub-packet size and SRQ on the network performance is discussed in Chapter VII.

B. NETWORK LAYER

The network layer performs routing functions to enable the transfer of data packets from a source to a destination via one or more nodes. It is responsible for setting up, maintaining and terminating connections and involves knowledge about the structure of the network. A topology defines how end points of a network are interconnected and how data flows. Optimizing the topology is essential in terms of capacity, energy consumption and reliability of the network. We focus our discussion of the network layer on suitable topologies to describe the Seastar network structure and data flow.

Some common basic topologies are bus, star, ring and tree [22]. Seaweb is normally structured with a tree topology. For Seastar, we narrow our candidates to the star and ring topologies, although hybrid forms might be options as well.

A star topology typically connects all nodes to a common central node. This central node acts either as a hub that collects and fuses information that is received, or it acts as a switching device where it relays information from one node to another. In a ring topology the network consists of a set of repeaters connected by point-to-point links forming a closed loop. Information flows in any or both directions. Both topologies have

one crucial shortcoming: a single point of failure at the central node for the star or any of the nodes in a ring. We cannot tolerate full network failure and mechanisms will need to be implemented to avoid it.

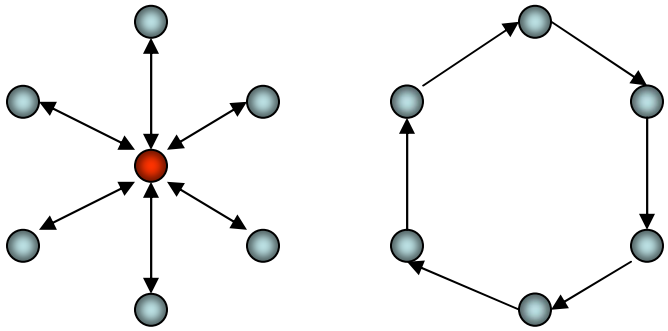

Figure 25 Star versus ring topology

1. Star Topology

For Seastar to operate in conjunction with Seaweb, the star topology would function as follows. The central node receives information from the peripheral nodes, fuses it and sends it as an information packet through Seaweb. The central node also behaves as a local command-and-control (C2) node for Seastar. A polling mechanism serves to avoid packet collisions, and error correction in the form of SRQ is issued as necessary from the central node to the peripheral nodes. The polling mechanism, which consists of a short utility packet containing address information, invites peripheral nodes to transmit data if available. Control information such as preferred output level or bit rate could be included. The downside of this mechanism is that it introduces overhead but it makes unnecessary the need for handshaking (RTS-CTS) and explicit acknowledgements (ACK). None of the peripheral nodes need to receive information from other peripheral nodes which simplifies both the network logic as well as the modem hardware. In order to reduce the complexity of the peripheral modems even further, the brief C2 messages

are transmitted at relatively low data rate and are reliably received with a simple demodulator. Conversely, the data transmissions from peripheral to central node are done at high bit rates.

The star topology is less susceptible to network failure than the ring. The central node, required to run the network, is its weak point and unfortunately the only way to avoid full network failure in case of central node malfunction is to have a backup node available in the network that could assume its duty. In the close presence of multiple clusters this may be achieved by reassigning the peripheral nodes to neighboring LANs. Another option is to have a mobile node available that could replace the failed central modem. In summary, backup options would either require significant advance planning or redeployment of spare hardware.

Once the choice for a topology is made, it is necessary to determine the most efficient strategy to operate the network. This is done with the aid of a simulation tool developed for this purpose and which is documented in Chapter VI. Early simulation experiments narrowed the number of strategy options for a star topology down to two. Both are described in more detail now.

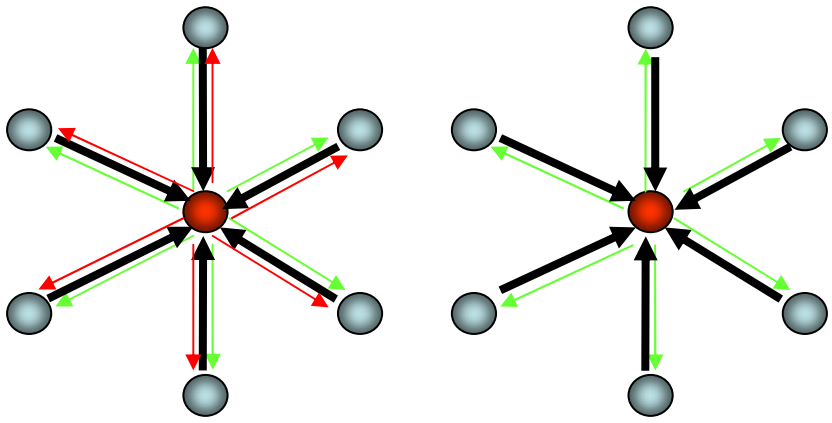

Figure 26 Candidate Seastar star topology strategies are P1D (left), which allows SRQ expressed by red arrows and P1E (right) that does not use this error correction feature. Poll and data transmissions are expressed by green and black arrows, respectively.

a. Star Topology Strategy Type P1D

Strategy P1D, as it shall be defined here, is graphically explained in Figure 27. It uses a short utility message, represented in the figures by a green arrow, which is transmitted omnidirectionally from the central node to poll a specific peripheral node. Upon reception, this node replies by omnidirectionally transmitting its data, preceded by a header containing information regarding the contents of the message such as source address, sequence number, message length and number of sub-packets. If the CRC of a specific sub-packet fails, an SRQ, represented in the figures by a red arrow, is initiated and, if necessary, repeated until all sub-packets have been successfully received or the maximum number of SRQ retries has been reached. Once the full data packet has been received by the central node it processes it and polls the next modem. The polling will continue uninterrupted.

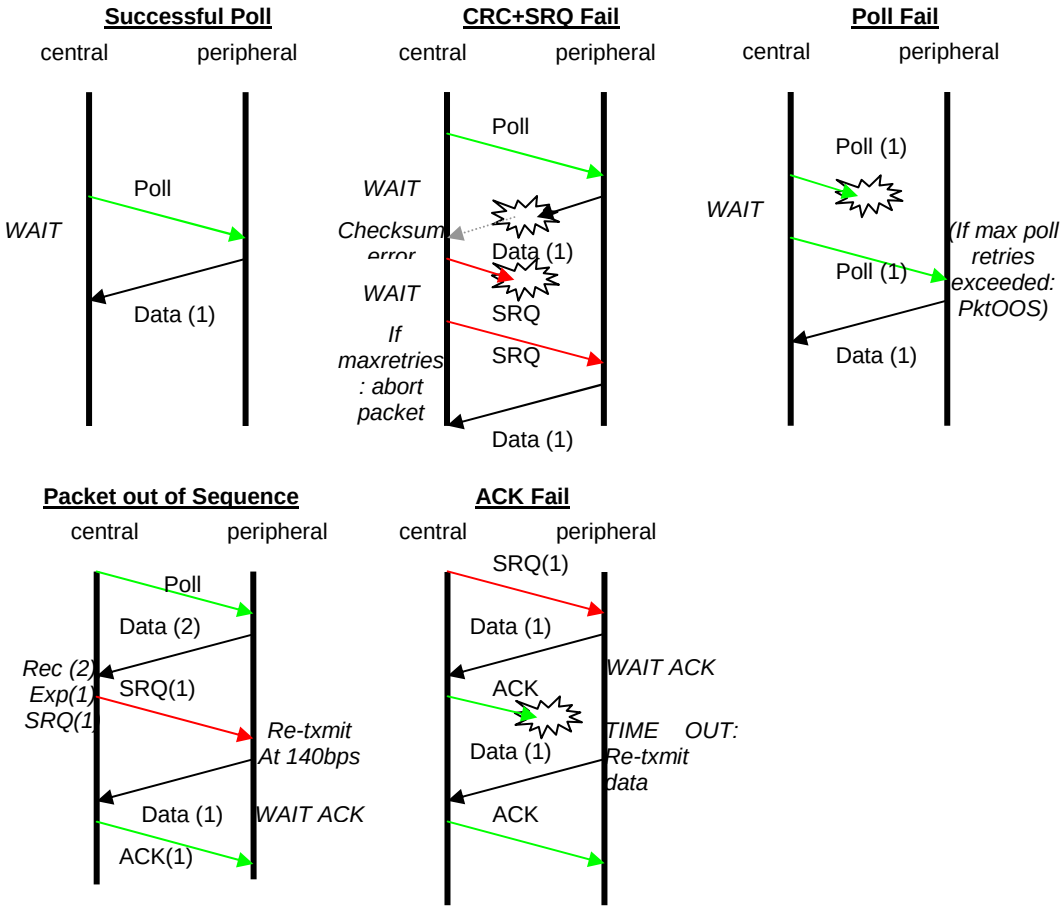

Figure 27 Graphical explanation of the P1D strategy as described in IV.B.1.a.

If the poll or SRQ utility packet is corrupted, no data transmission ensues and a time-out period for reception at the central node indicates that something has gone wrong. In this case a retransmission of the utility packet is issued until either the data packet is received or a maximum number of SRQ or poll retries has been reached. In the case of maximum SRQ, the packet is aborted and considered lost. In the case of a maximum achieved number of polls, the peripheral node maintains track of the data and its sequence number for transmission at the next cycle. At the next cycle, the choice can now be made for the central modem to either ask for the latest, most up-to-date sequence number (so implicitly aborting the previous number) or to have it issue an SRQ for full retransmission. In this last case, as well as in unforeseen situations where a packet ends up out of sequence, an explicit ACK for reception is issued by the central node.

b. Star Topology Strategy Type P1E

Strategy P1E is based on P1D but does not perform sub-packet recoveries. The motivation for this variation is to support network operations where low latency is a higher priority than transmission reliability, thus favoring a low amount of overhead. All corrupted packets are consequently aborted. P1E does, however, poll again upon failure but this is limited to one additional attempt. The description for P1E is therefore the same as for P1D but excludes SRQ.

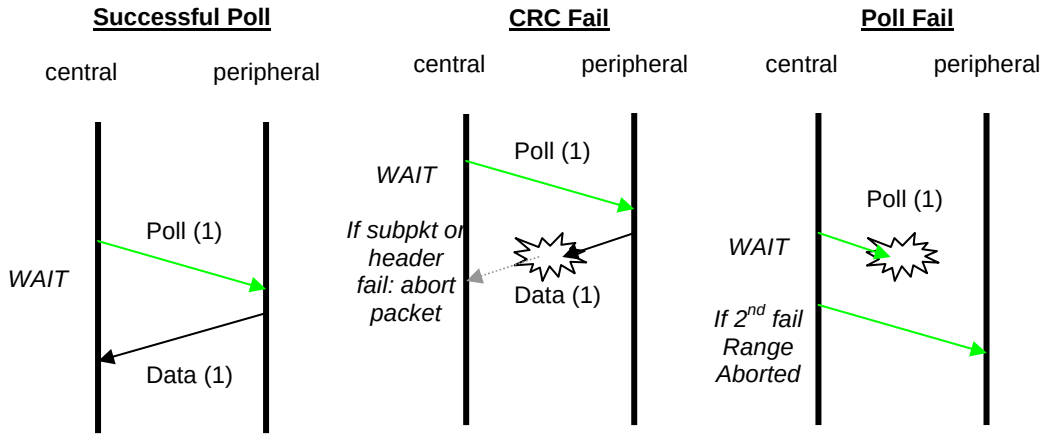

Figure 28 Graphical explanation of the P1E strategy as described in IV.B.1.b.

2. Ring Topology

Although we will refer to a ring topology, the idea actually diverges from the traditional ring since the central node is included. This is not only required to connect Seastar to Seaweb but it also provides the possibility to perform centralized C2 duties in case of network failure. The main difference with the star topology is the absence of a polling mechanism from the central modem. Instead, a token is passed between neighboring peripheral nodes without interruption from the central modem. Not only does this reduce overhead but it also reduces the energy consumption at the central modem. A peripheral node is only allowed to transmit data upon reception of a token that is received from the previous node in the cycle. Because the token is transmitted omnidirectionally, address knowledge between neighboring nodes is required and included in the token. Data packets are also transmitted omnidirectionally but processed by the central node only. The data transmission is padded by the updated token, which cues the next modem to transmit data. It is obvious that a ring topology requires not only communications between central and peripheral nodes but also between neighboring peripheral nodes, which makes the ring less suitable for independently moving nodes.

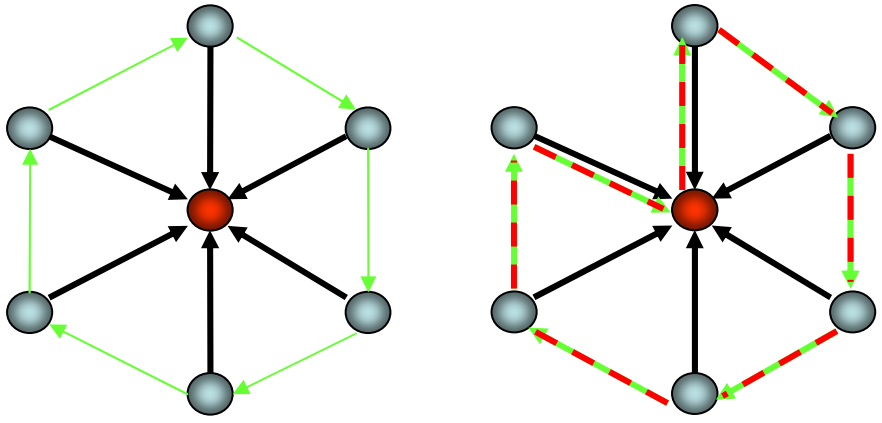

Figure 29 Candidate Seastar ring topology strategies are T2B (left), which does not use SRQ, and T3A (right), which allows SRQ as an integrated message within the token, updated by the central node every cycle. Token transmissions are expressed either as green or as red-green arrows and data transmissions as black arrows, respectively.

As with the star topology, RTS-CTS and ACK messages are not required but introducing SRQ or token retransmission is more complicated since C2 occurs on two levels. In case of a corrupted token, retransmission would have to be coordinated between neighboring nodes whereas corruption of sub-packets is handled between central and peripheral nodes. This complicates the network logic and forms the basis of creating two variations on the ring theme as described in the following subsections.

a. Ring Topology Strategy Type T2B

Strategy T2B involves passing a token amongst the peripheral nodes and can be compared to P1E in the sense that it does not provide SRQ. The central node receives and processes all packets that are received successfully and aborts all corrupted packets. In case of a corrupted token the network would normally fail completely. However, the central node will sense that no data is transmitted and will retransmit the token to the last expected address after a time-out period. If still no data packet is received the central node will reinitiate the token again but now it is addressed at the next peripheral node in the cycle. Figure 30 provides an overview of the network logic.

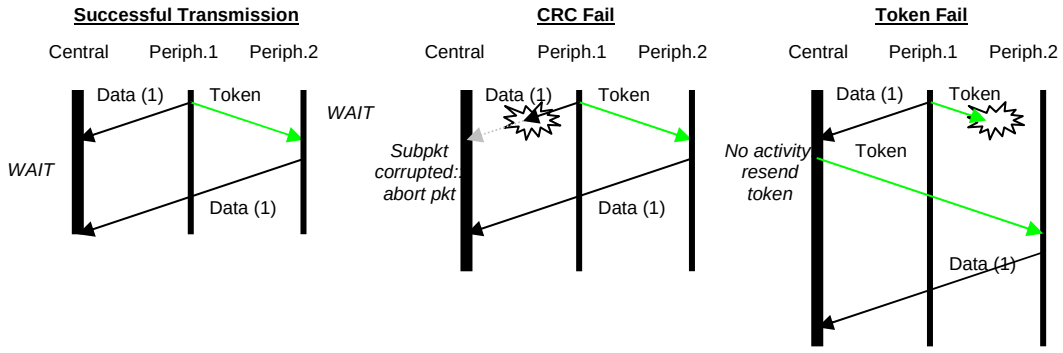

Figure 30 Graphical explanation of the T2B strategy as described in IV.B.2.a.

b. Ring Topology Strategy Type T3A

An alternative ring strategy introduces an additional hop in the ring by passing the token though the central node and is referred to as T3A. During a cycle, the

central node stores information regarding unsuccessful sub-packet transmission from each peripheral node. The token, now expressed as a red-green arrow, is designed to carry additional information or instructions, such as SRQ, and is updated by the central node. The token carries this information with it in the ring and delivers it at the target node. The length of the token utility packet therefore scales with the number of nodes in the networks but clever design can reduce the additional overhead. Upon reception of an SRQ, the peripheral node will both retransmit corrupted (sub-)packets and transmit new data in the same cycle. Unsuccessful retransmissions will generate no new retransmission because of long latencies for that specific message and the full packet will be aborted. The same time-out logic as with T2B is implemented in case of token failure. Figure 31 summarizes the logic of T3A.

Note that with the graphical descriptions of P1D and T3A, an unsuccessful poll or token generates a packet-out-of-sequence situation. This just occurs for simulation purposes. Actual implementation could include either packet abortion or retransmission in the subsequent cycle.

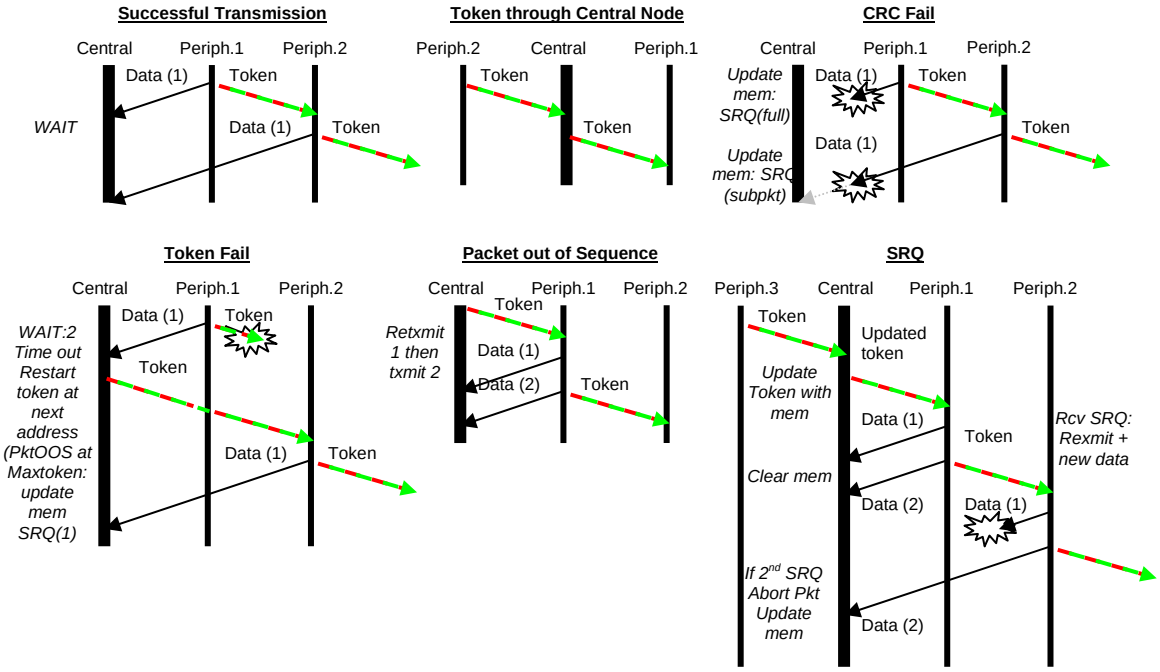

Figure 31 Graphical explanation of the T3A strategy as described in IV.B.2.b.

V. SEASTAR PROTOTYPE IMPLEMENTATION AND SEA TESTING

A. PROTOTYPE IMPLEMENTATION

A Seastar prototype was developed to test the concept of a centralized network with through-water acoustic links. The following goals formed the basis for the Seastar prototype:

- Verify suitability of asymmetric acoustic links in air and water,
- Develop a prototype network, demonstrate the feasibility in air and provide initial performance metrics,
- Demonstrate the feasibility in water and provide quantitative and qualitative analysis.

The first two goals were achieved by performing experiments in the anechoic chamber and the anechoic water tanks at NPS. The last goal was achieved by an experiment as part of AUV Fest 2007 in Panama City, FL.

1. Asymmetric Link Experiment

For the asymmetric link experiment, a commercially available Teledyne Benthos ATM-885 subsea modem, an ATM-891 deck box and an AT-408 omnidirectional transducer were used. Both the ATM-885 and ATM-891 were uploaded with standard commercial firmware version 5.5. The modems operate in the 9–14 kHz band and are designed to communicate over distances up to 5 km. The modulation type can be set to either multi-channel MFSK with bit rates varying from 140 bits/s to 2400 bits/s or PSK with bit rates varying from 2560 bits/s to 15360 bits/s. Transmit power levels or source levels can be set at 164–185 dB re 1 μPa @ 1 m in water which corresponds to 102–123 dB re 20 μPa @ 1 m in air. Communication ranges in the anechoic water tank varied from 10 cm to 4 m whereas ranges in the anechoic chamber varied from 10 cm to 1 m. Interaction with both the ATM-885 as well as the AT-408 through the deck box was

established by connecting the RS-232 feeds on the modems to two USB ports on an HP Pavilion DV5000 laptop computer. IOGEAR USB-to-serial adapters were used to connect the RS-232 devices to the USB ports. Symantec Procomm Plus provided a graphical user interface (GUI) to interact with the modems.

One interpretation of an asymmetric link was tested easily. Transmitting short (9-byte) control messages from the deck box to the ATM-885 modem followed by a long (up to 4096-byte) message reply was trivial. The next step was to test asymmetry with respect to bit rate. Both in air and water, transmitting at the lowest available bit rate (140 bits/s) and replying at the highest possible bit rate for MFSK (2400 bits/s) produced few problems although some transmissions failed at 2400 bits/s. Transmitting messages at lower bit rates (up to 1200 bits/s) using FEC and coding was never a problem, which demonstrates the benefits of applying error correction techniques.

The last asymmetry that was tested involved further increment of the bit rate by switching to PSK. Some occasional transmission successes at 2560 bits/s were achieved but this was hardly enough to make it feasible for practical application. Higher PSK bit rates with this setup failed in every attempt.

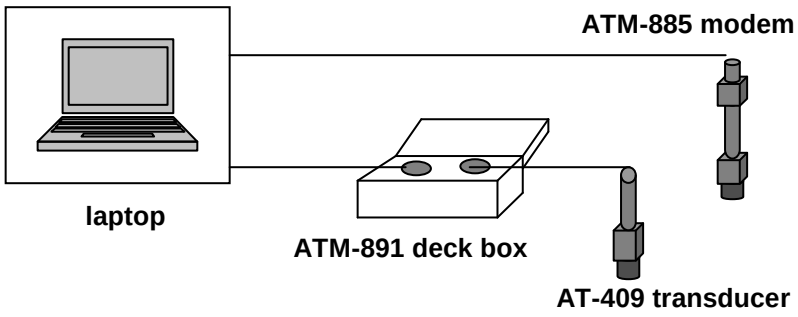

Figure 32 Experimental setup for asymmetry tests as described in this section.

The overall conclusion of the first experiment is that asymmetric MFSK transmission using commercially available modems is practical for Seastar prototype purposes. However, transmitting at even higher bit rates using PSK modulation could not be achieved with the same equipment and was therefore found not suitable for prototype implementation.

2. Seastar Prototype Development in Air

Once the asymmetry possibilities were known, the Seastar prototype could proceed. The ATM-891 deck box and AT-408 transducer combination represented the central node and five ATM-885 modems served as peripheral nodes. An unsuccessful attempt was made to use a Brüel and Kjær PULSE analyzer for impulse response and frequency-time measurements and at the same time have it function as a tool to trigger recordings by means of a matched filter. The equipment was later replaced by a G.R.A.S.-type 40AF free field microphone and type 26AK 1/2" pre-amplifier combination, connected to the laptop and powered by a G.R.A.S.-type 12AA module. The application that was used to perform the time-frequency spectrum recordings and analysis was Spectrogram version 15.1.[1]

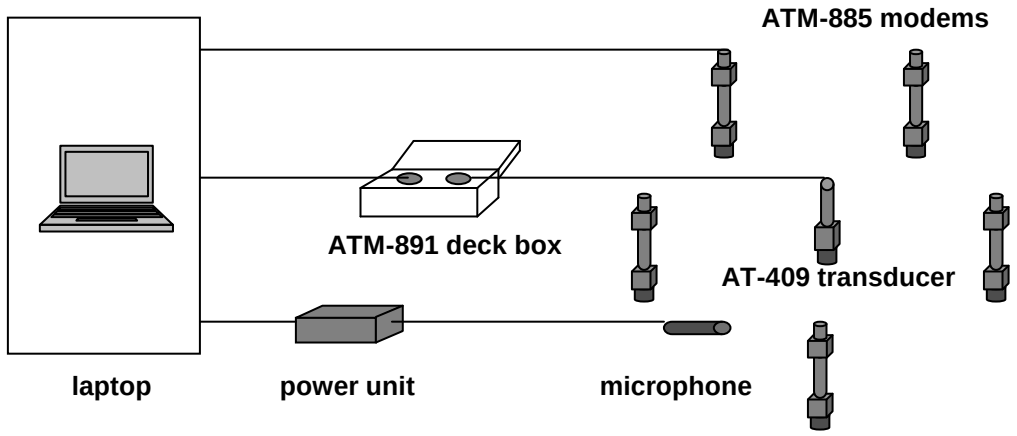

Figure 33 Experimental setup for Seastar prototype in anechoic chamber as described in this section.

In a later phase of the experiment the AT-408 transducer was replaced by one of the peripheral modems as can be seen in Figure 34. This replacement had no further impact on the experiment. The laptop was always connected to the central node and one of the peripheral nodes and served as a GUI to send command messages, manipulate modem and network settings, and provide real-time feedback on transmissions.

[1] Spectogram v. 15.1, Visualization Software LLC, http://www.visualizationsoftware.com/gram.html, Accessed 2 December 2007.

The topology that was most easy to create with the available equipment was a star topology. A ring topology would have required software modification in the modems. The first step was to determine a way to implement a polling mechanism. Fortunately an existing 9-byte utility packet was found suitable to perform this function. The ASCII command "AT$BT*n*", where *n* refers to a modem address was originally developed to acoustically order modem address *n* to transmit the contents of its data buffer. The data buffer is usually filled with data from a sensor that is hooked up to the serial port. For our purposes, an 1850-byte test message was manually uploaded and stored in the buffer of all peripheral modems and resided there until it was manually erased.

Figure 34 Seastar prototype setup in NPS anechoic chamber showing one central node and four peripheral nodes. The range between the peripheral nodes is approximately 1.5 m.

We now had a 9-byte polling message (AT$BT*n*) to poll any modem and have it respond by transmitting an 1850-byte data packet that was divided into eight 256-byte sub-packets, a Benthos modem feature. To introduce the asymmetric link, the poll was set to be transmitted at 140 bits/s, followed by a data packet transmission at 800 bits/s. The next step was to automate the polling mechanism. This required an algorithm that had to run from an external CPU that was connected to the central modem through the RS232 connection. Since the algorithm had to be installed on a UNIX-like driven CPU during

the follow-on experiment at sea, the choice was made to exchange the laptop computer for a Linux machine. Since we abandoned the PSK modulation, there was no need to continue working with the commercial code and all modems were uploaded with the Seaweb source code, version 17.3, providing the modems with extended network features, such as SRQ, that would be useful in our prototype Seastar implementation.

The C algorithm, as shown in Appendix A, is a modification to the original software used on the Seaweb Racom (radio/acoustic communications) gateway buoys to allow interaction with the Seaweb network. The polling algorithm includes several recovery features and additional delays to prevent network failure and performs an automatic restart in case of a full network crash. The challenge lay in the fact that the newly developed polling algorithm needed to work in conjunction with the existing Seaweb modem firmware. For example, the polling has to be suspended whenever a transmission is corrupted to allow SRQ. Upon successful transmission, the polling mechanism must then automatically retake control and continue the polling cycle.

Handshaking through RTS-CTS as well as explicit acknowledgements through ACK utility packets was disabled with SRQ enabled by setting the modem's S-registers as follows: S33=3, S34=0, S57=0. With these settings, handshaking only occurs upon network initialization and whenever the central modem comes out of low power state. A 10-second delay after data transmission was built in to avoid overlap of polling and retransmission of full packets in case of a packet-out-of-sequence situation. As a final recovery mechanism, a 10-minute timer was inserted in the code to enable an automatic network restart in case of full network failure.

Upon detection of multiple simultaneous transmissions, the central node ceases polling for 10 minutes to allow all modems to assume a low-power state and, in doing so, clear all sequence-number memories. The polling, preceded by a new handshake, automatically resumes after this silent period. A spectrogram of a single round-trip transmission containing a poll and a data reply can be seen in Figure 35.

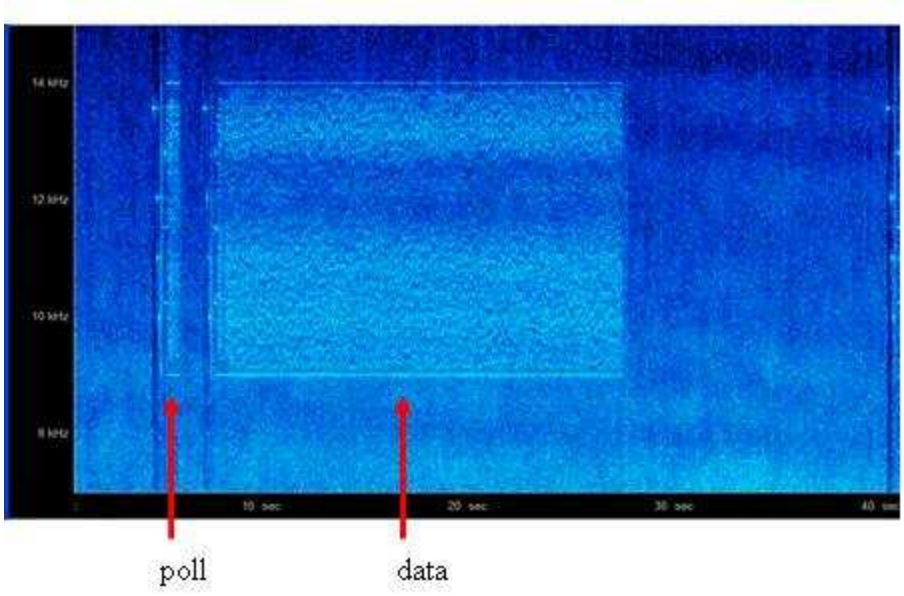

Figure 35 Time (horizontal axis) and frequency (vertical axis) recording in water of 9-byte poll at 140 bits/s followed by 1850-byte data transmission at 800 bits/s.

With these settings, the network is able to operate under the influence of physical-layer faults. We will refer to a single round-trip-time (RTT) as the time required for transmitting a poll followed by a data transmission including delays. An average single RTT without retransmissions was measured to be 34.5 seconds. The exact RTT depended on random delays that are introduced by the Benthos modems as a built-in feature but the deviation was never more than 3% of the average value. For five peripheral modems this would mean that the average cycle time would be 173 s. This value was found to be a useful metric since it indicates the mean time between data transmissions from a particular Seastar modem. We will further refer to the cycle time as *latency* with units of seconds. The latency is also affected by the amount of retransmissions required and therefore implicitly indicates the reliability of a topology. If SRQ is disabled, however, the latency will remain constant at an unsuccessful transmission but the reliability drops. A second metric that does account for this and can easily be measured is the *number of dropped packets*. It is, however, desirable to evaluate this number independent of the amount of packets that were transmitted. We therefore normalize the number of packets by expressing it as a ratio of the number of unsuccessfully transmitted packets to the total

number of packets as a percentage. From a network efficiency perspective, it is useful to know the efficiency or *utilization* of the channel. This third metric is defined as the ratio of time T_i required to transmit information bits over the total time T_t required to perform this transmission. The total time includes overhead caused by headers, CRC, redundancy, retransmissions, and necessary polling or token utility packets. As an example, T_i for 1850 bytes at 800 bits/s would be 18.5 s, but as was shown before, T_t is 34.5 s. The dimensionless utilization is therefore 18.5 s divided by 34.5 s which is 0.536. In other words, only 54% of the channel availability was efficiently used to transmit information. The last metric is related to the channel utilization but expresses the efficiency in a more operational sense. It is defined as the number of transmitted information bits per RTT. This metric is further referred to as *information throughput* and has units of bits/s. As an example we will use the same data as above and determine the information throughput for our experimental setup to be 8 bits/byte times 1850 bytes divided by 34.5 s which is 430 bits/s. In other words, although 800 bits/s was the physical-layer bit rate, this specific experimental setup only achieves a maximum network-layer bit rate of 430 bits/s, which is again 54%.

Conclusively we can state that a first Seastar prototype, using a polling mechanism, was successfully developed, tested in air and analyzed. The analytical metrics that were found useful are: information throughput, channel utilization, latency, and dropped packets. These metrics are used throughout the rest of this research. For our in-air experimental network, the following values were found: information throughput 430 bits/s, utilization 0.536, latency 173 s and a zero percentage of dropped packets, recognizing that these values were obtained under almost perfect test conditions by using the anechoic chamber. Now, we analyze the performance of this version of the Seastar prototype while deployed in realistic conditions at sea.

B. EXPERIMENT PLAN

In June 2007, a Seastar prototype was tested in water during the AUV Fest demonstration at St. Andrews Bay, FL. The goal was to demonstrate the feasibility in water of the prototype described in the previous section and provide a quantitative and

qualitative analysis. The plan consisted of deploying the network in shallow water in a moderate shipping area to observe influences of natural and man-made interference on the network performance. To measure network performance quantitatively in metrics of information throughput, channel utilization, latency and dropped packets it was necessary to use equipment capable of recording number and status of received poll and data packets as well as SRQ utility packets, all tagged with time stamps. For a qualitative analysis it was required to associate the successful and anticipated unsuccessful transmissions to the channel conditions, possible noise sources, and network settings. Direct access to the network to manipulate settings and observe related performance would allow additional quantitative and qualitative analysis data and could provide calibration data for future network simulations.

The available hardware consisted of five Teledyne Benthos ATM-885 modems that would serve as peripheral nodes and a central Racom gateway buoy that would perform the function of central node and gateway to the Seastar network. To achieve this, the Racom was equipped with a Teledyne Benthos ATM-885RPCB modem board, an AT-408 omnidirectional transducer, Iridium satellite communications and FreeWave radio. It further contained a central processing unit (CPU) that was used to upload the polling algorithm as well as the original algorithm to enable manual network access. Not only did this allow changing network parameters but it also permitted troubleshooting and uploading the C program in case debugging of the code was required. Last, the Racom allowed local storage of network data which formed a backup in case of radio communications failure. Remote monitoring of the network would occur from a Seaweb server [39] at Naval Surface Warfare Center (NSWC) Panama City, FL. To ensure qualitative analysis, a moored sonobuoy,[2] capable of recording raw acoustic data such as network transmissions, shipping, and other interference (e.g., see [40]), would be deployed within 200 m of the central modem. The data recorded by the sonobuoy were transmitted to a laptop ashore and could be analyzed real time by the Spectrogram application. Analysis was further informed by conductivity-temperature-density (CTD)

[2] SeaLandAire Technologies, Inc. http://www.sealandaire.com/currenptojects.php. Accessed 2 Decemeber 2007.

profiles and other local data at the experiment site, such as wind and visual and/or surface radar tracks, that were collected both by SPAWAR Systems Center San Diego and US Navy METOC personnel.

Figure 36 Upper picture shows a peripheral modem attached to a weight, acoustic release and a floating body for vertical positioning and recovery. Lower pictures show the sonobuoy (left) and Racom buoy (right).

C. EXPERIMENT SETUP

1. St. Andrews Bay

St. Andrews Bay is connected to the Gulf of Mexico and is part of the intra-coastal waterway system. The Seastar test site was located 1 km east-southeast of the main commercial port, as can be seen in Figure 37. The water depths at the site vary from 8 m to 13 m and the bottom consists of an acoustically absorptive mud/silt composition. Surface temperatures during the experiment were generally over 30 degrees Celsius and a moderate southwest breeze usually developed in the afternoon causing an average sea state of 1 (0-0.1 m).

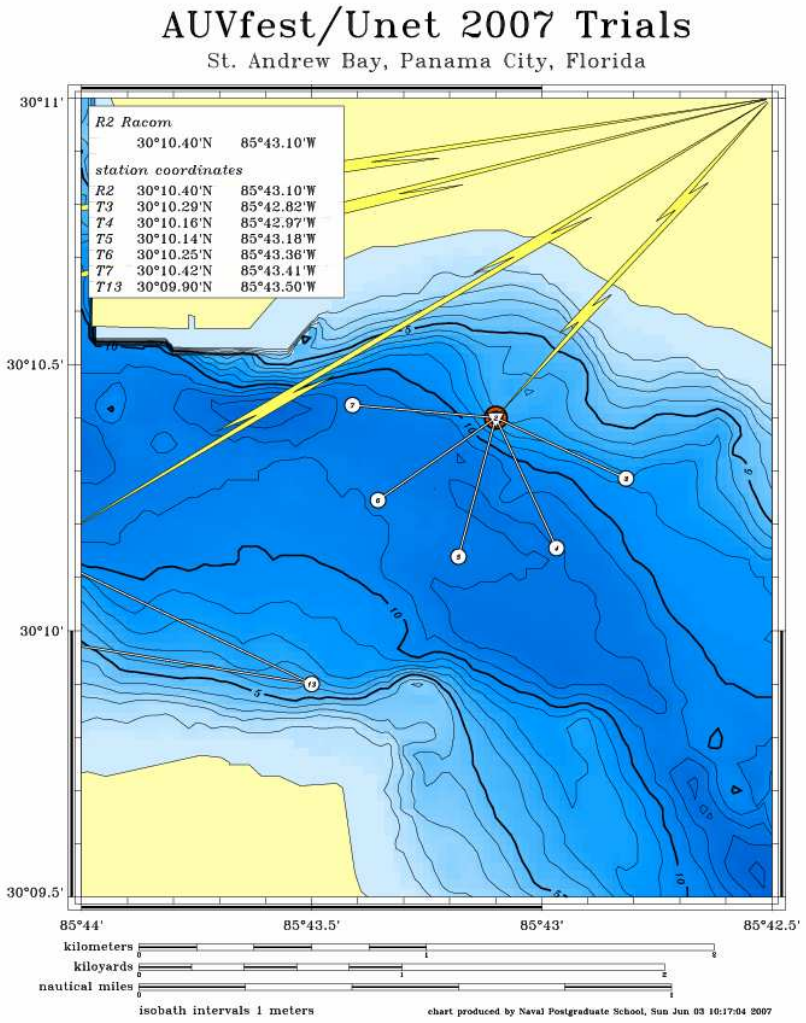

Figure 37 Geographic overview of AUV Fest / UNET test site showing the Seastar prototype network geometry in combination with the depth contours in meters. Panama City's main port lies 1 km west-northwest of the central node.

Occasional tropical rain showers causing severe variations in the sound velocity profile were expected; however, none occurred during the actual data collecting phase. Two series of CTDs taken prior and during deployment resulted in sound-speed profiles

(SSPs) as shown in Figure 38. The absence of heavy rain and wind during the week caused the SSP at the test site to remain stable although a negative gradient developed near the surface.

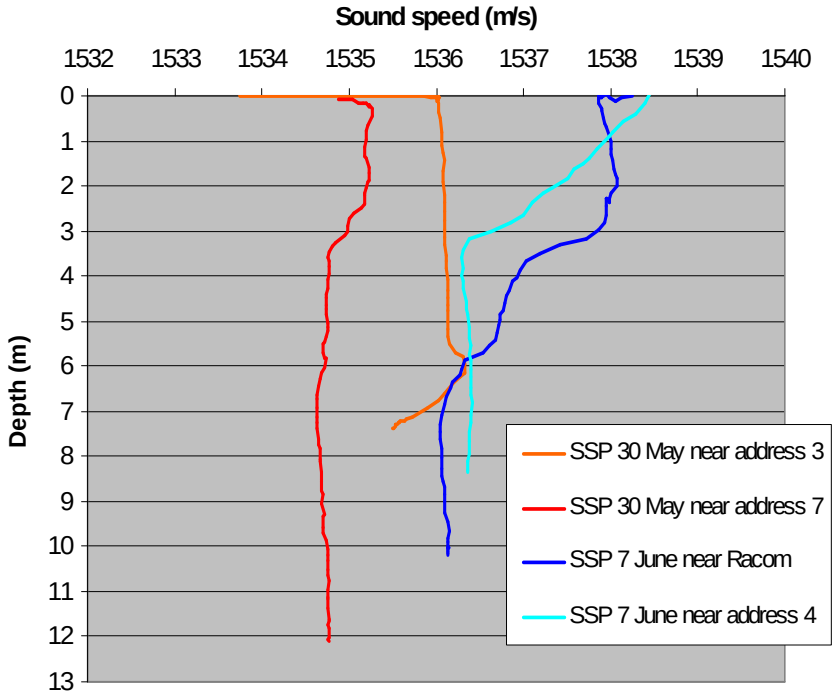

Figure 38 SSPs taken at the test site show only a slight increase in temperature over a period of a week and the development of a negative gradient near the surface.

Sound propagation predictions based on the SSPs for June 7, one of the actual data collection days, are presented in Figure 39. This figure was generated by a Matlab application developed by Torres [41] that uses the Bellhop Gaussian beam tracing acoustic propagation model. Torres demonstrated that Bellhop is suitable for modeling high-frequency acoustic propagation in shallow water and performed several case studies for St. Andrews Bay. Even though our experiment used medium frequencies, the model provided useful data for determining the most suitable deployment depth for the modems to ensure communications. The Bellhop model shows a downward refracting communications channel and surface and bottom bounce paths. The almost isospeed channel also supports direct-path propagation, which is most favorable since it

experiences the least transmission loss of all multi-paths. The bottom-surface interactions induce expected multi-path time dispersion of 23 ms.

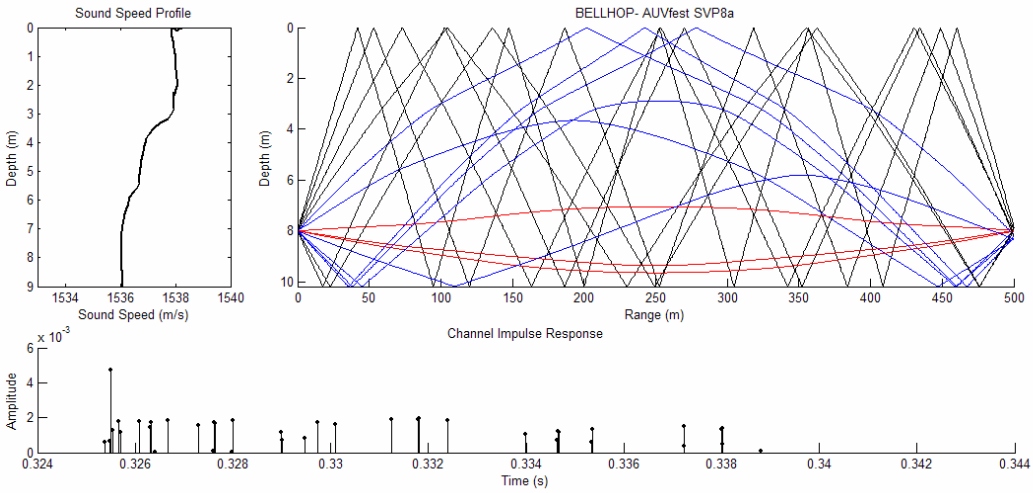

Figure 39 Bellhop predictions for 7 June at a location between the Racom and T7 show a downward refracting communications channel that allows direct path. Multi-path arrivals due to bottom and surface interactions are also expected.

The average multi-path delay as measured by the modems was 0.72 ms. Compared to the predicted maximum of 23 ms, we can conclude that the main propagation path during the experiment was direct path and interference due to multi-path delay is not considered to have been a significant factor. The nearly stationary, almost isospeed channel conditions therefore made the environment at the test site well suited for underwater acoustic communications.

Noise sources consisted mainly of shipping, wind and sea life. Shipping noise was episodic, arising from small private vessels and occasional commercial vessels. Traffic associated with AUV Fest also contributed to shipping noise. The most significant noise source in our operating band, however, appeared to be interference from other experiments in the bay that used Teledyne Benthos modems.

The maximum observed wind during the actual data taking was 15 knots which, according to the Wenz curves [17], leads to ambient noise levels of 47–50 dB on the 9–

14 kHz band. Although the effect of noise produced by shipping generally diminishes above 1 kHz as shown in Figure 6, interference from vessels crossing the network at close range was observed.

Noise generated by shrimps and other sea life was observed and recorded but no causal interference could be verified during the trials.

Variability in the noise was mainly dependent on the activity of the other Seaweb experiments being conducted in the bay and was not noticeably influenced by natural conditions at all. Overall, the propagation and noise conditions were favorable for testing the Seastar prototype network.

2. Network Setup

All five peripheral modems were deployed at ranges of 500 m from the central node (Figure 37) causing an average one-way propagation delay of 0.3 s. The geometry of the channel and presence of other networks in the vicinity did not allow a symmetric setup but this was not a requirement for the polling strategy. In accordance with the Bellhop propagation predictions, the modem transducers were positioned at 3 m from the bottom. The transmit power levels of the central node and peripherals were set to 179 dB re 1 μPa @ 1 m. The acoustic baud rates of the peripheral nodes and central node were set to 800 bits/s and 140 bits/s, respectively. The poll and data message as well as the SRQ, RTS-CTS and ACK settings (S33=3, S34=0, S57=0) remained unchanged from the in-air experiment. Based on the 500-m range and the in-air experiments, an average RTT of approximately 35 s was expected.

As stated before, the control of the network and data recording were supposed to occur remotely using a server ashore. However, technical problems with both the Iridium modem and the FreeWave modem left local recording on the Racom as the only option. This also implied that the network could not be adjusted or manipulated once it was deployed, which limited the scope of the experiment. All tests were therefore autonomously conducted with the above settings.

D. NETWORK EVALUATION

Two trials were performed. Trial 1 started on June 7 at 12:00LT and lasted until June 8 07:30LT. Trial 2 started on June 8 at 10:15LT and ended at 16:30LT that same day. Trial 2 included a controlled run by a small boat over the network. Both trials started with a failure from unknown causes, which activated the 10-minute out-of-action period that was hard-coded in the polling algorithm. It was observed that the intended 10-minute silent period lasted almost two hours. We hypothesize that interference from the adjacent Seaweb network that used similar modems is the cause for this unexpected behavior. The activity of these modems prevented the Seastar modems from going into a low-power state, which made a fresh restart impossible. Once the activity of the other network had ceased, the modems entered a low-power state and the restart occurred as intended.

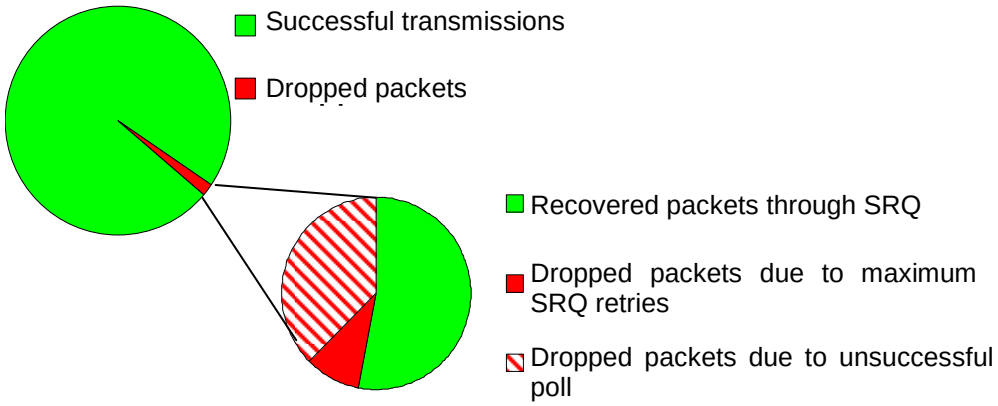

Figure 40 Summary of Seastar prototype performance in water where 99.3% of the transmissions were successful.

During these trials, where 26 hours of network operation was achieved, a total of 2031 successful data transmissions were made by the peripheral nodes (see Figure 40). Only 17 packets were initially unsuccessful, but 14 of these were recovered through SRQ. An additional 12 packets were dropped because the poll was never received. The total number of sub-packets that were corrupted was 53. The relevance of this number can be found when considering selective retransmissions through SRQ versus full

retransmission if selectivity had not been used. Instead of 34816 bytes, only 13568 bytes had to be retransmitted, which is a reduction of almost 60%. Only three packets were found out of sequence and they were all retransmitted successfully. Including the network failures at startup, a total of three full network self-recoveries occurred and human intervention in the network was never required.

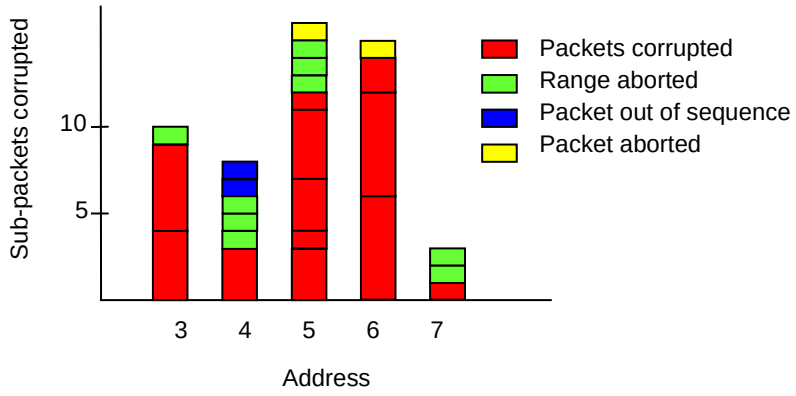

Figure 41 Summary of unsuccessful transmissions during Trial 1 for Addresses 3–7. Addresses 4, 5 and 6 experienced the most interference.

Trial 1					
Address	**3**	**4**	**5**	**6**	**7**
Sub-packets Corrupted	4 5	3	3 1 3 4 1	5 5 2	1
Polls corrupted (missed packet)	1	3	3	1	2
Packets out of Sequence	-	2	-	-	-
Packets aborted due to maximum SRQ	-	-	1	1	-

Table 3 Summary of amount and description of unsuccessful transmissions during Trial 1 for Addresses 3–7.

The percentage of successfully transmitted packets was 99.3%. A summary of unsuccessful transmissions is found in Figure 41 and Figure 42 as well as in Table 3 and Table 4.

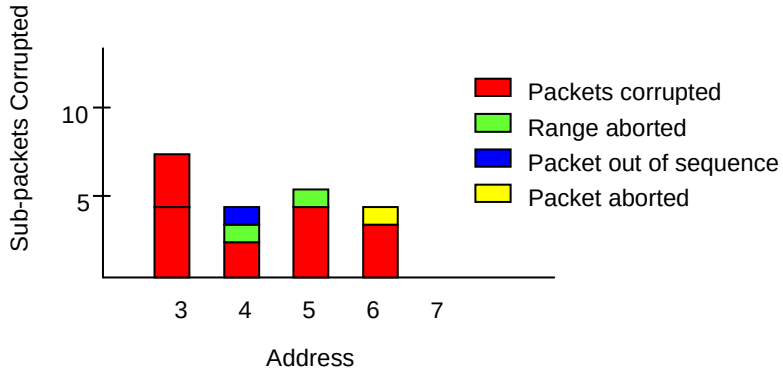

Figure 42 Summary of unsuccessful transmissions during Trial 2 for Addresses 3–7. The number of unsuccessful transmissions was too few to be of statistical significance but the trend is similar to Trial 1.

Trial 2					
Address	**3**	**4**	**5**	**6**	**7**
Sub-packets corrupted	4 3	2	4	3	-
Polls corrupted (missed packet)	-	1	1	-	-
Packets out of sequence	-	1	-	-	-
Packets aborted due to maximum SRQ	-	-	-	1	-

Table 4 Summary of amount and description of unsuccessful transmissions during Trial 2 for Addresses 3–7.

We now express the performance of the Seastar prototype network in water under the above-described conditions in terms of the metrics used during the in-air trials. The average latency in water, based on the data in Figure 43, Figure 44 and Table 5, is 181 s compared to 173 s during the in-air experiment. Based on the latency, an information throughput of 408 bits/s is found with a channel utilization of 0.51. The percentage of dropped packets is 0.7%. Only a small portion of this performance degradation can be

attributed to the longer propagation delays. The major contribution comes from the fact that unsuccessful transmissions and retransmissions caused additional delays.

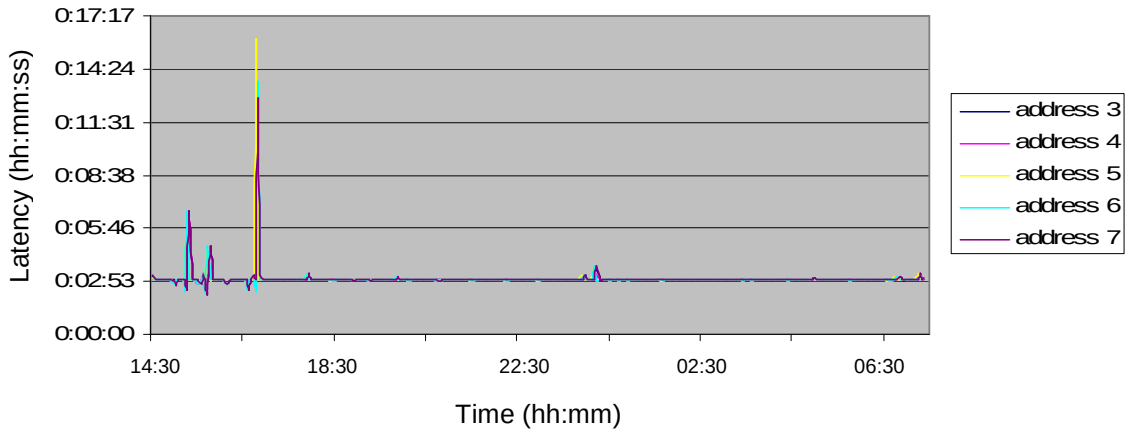

Figure 43 Latency (vertical axis) measured during Trial 1. The peaks are due to either long retransmissions or full network restart.

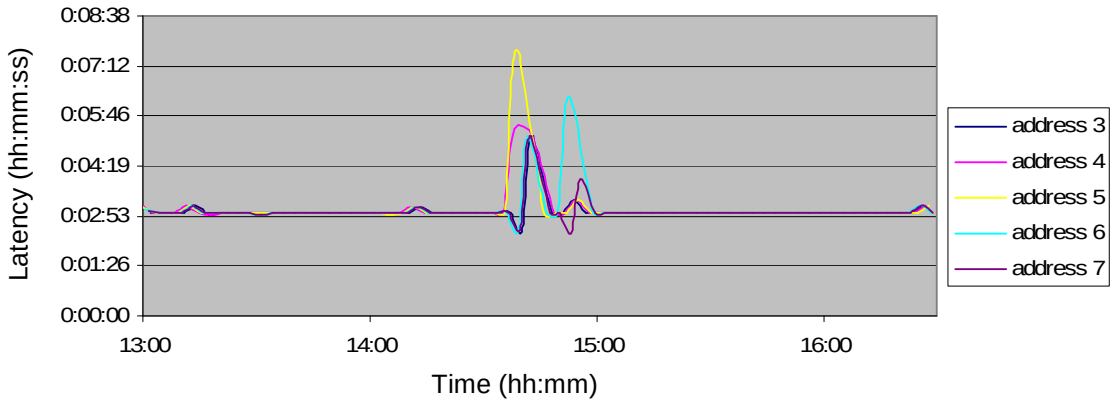

Figure 44 Latency (vertical axis) measured during Trial 2. The peaks are due to long retransmissions.

Address	**Trial 1 h:mm:ss**	**Trial 2 h:mm:ss**
3	0:03:01	0:03:00
4	0:03:01	0:03:03
5	0:03:01	0:03:03
6	0:03:01	0:03:03
7	0:03:01	0:03:00

Table 5 Average latency for specific modems during both trials.

From a qualitative perspective, we would like to associate the unsuccessful transmissions with certain events or conditions. The sonobuoy was a superb tool for identifying interference sources. Four unsuccessful transmissions occurred due to the passage of a small boat, another four were caused by a large boat and four transmissions were corrupted due to interference from the nearby Seaweb network, although it must be mentioned that most of the Seaweb transmissions did not interfere with Seastar operations. The cause of one unsuccessful transmission could not be determined. All other failures occurred at time intervals during which the sonobuoy data were not logged. Recordings of marine mammals' sonar occurring in the Seastar band did not indicate any negative interference. An example of an SRQ and data retransmission due to the passage of a small boat is shown in Figure 45.

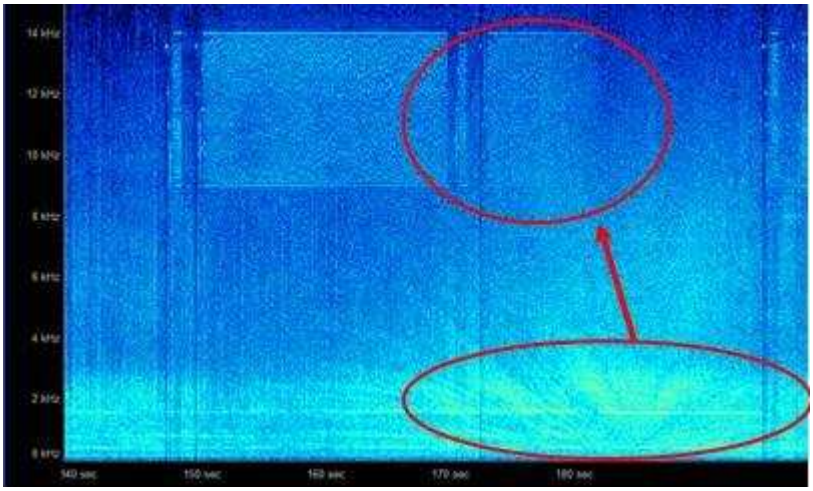

Figure 45 Interference due to passage of small boat causing SRQ.

Addresses 4, 5 and 6 required the most retransmissions and the majority of the dropped packets also originated from these addresses. Knowing the causal relation between interference from both shipping and the Seaweb modems, it is not a surprise to find that these addresses are both in the shipping channel and in close proximity to the Seaweb network. On the other hand, no reasons can be found for the exceptionally good performance of Address 7, since it was closest to the port facilities and also close to the Seaweb network.

We conclude this chapter and the experiment sequence with the following statements. The Seastar prototype has been successful in air and sea trials. The star

topology in combination with a polling mechanism has proven to be a robust strategy that is able to operate autonomously for a long period. It needs to be mentioned that we were not able to manipulate network parameters and that the results were obtained under favorable conditions. Although interference from shipping and Seaweb was observed, we must take into account the fact that Seastar will be operating in a higher frequency band. Since the anticipated operational environment has similarities with the test site, further testing with future high-frequency modems in the same environment is strongly advised. Although the network was reliable, it was found that the performance was limited. The low data rates in combination with the polling (TDMA) strategy resulted in relatively high latencies and a low information throughput. This was mainly due to the built-in random delays and the additional 10 s delay that was required to ensure smooth cooperation between the Seaweb software and the C polling algorithm. Smart integration of both algorithms and reducing the delays in the hardware could greatly increase the network performance. Further gains may be realized by optimizing the network strategy. Since existing hardware does not allow easy implementation of strategies such as P1D, P1E, T2A and T3A, this will be done in the next chapter though simulation.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. NETWORK SIMULATION

Having demonstrated a Seastar prototype using available Seaweb equipment, we now explore various networking strategies by simulating them on a computer. The simulation allows us to analyze, evaluate, and optimize various candidate strategies. The output metrics of the simulations are presented in terms of channel utilization, information throughput, latency, and dropped packets. The results of the experiments with the prototype network described in Chapter V provide a perfomance benchmark.

In the first section of this chapter we discuss the setup of the simulation. The next section shows parametric results for the four network strategies (P1D, P1E, T2B and T3A) described in Chapter IV. We conclude with a summary of pros and cons for these strategies. This then forms the basis of case studies that are performed in Chapter VII.

A. SIMULATION SETUP

Matlab source code for the network simulation is fully shown in Appendix B. We designed the simulation to provide the possibility of analyzing multiple network types simultaneously under similar conditions. Network parameters, to be described soon, can be set either as single value or as an array of values. The simulation is time and event driven and depends on random processes to trigger certain events, such as failure of a transmission. In order to generate statistically relevant results where the effect of outliers is insignificant, each simulation is repeated a large number of times. The simulation duration as well as the number of simulation repeats can be set manually. This is also the case for the SNR threshold levels above or below which a certain event will occur. Even though these levels are arbitrary, the events that are triggered behave like they are caused by noise so that the effects of performance degradation on the communctaions can be studied. Each networking strategy is evaluated simultaneously under the same conditions and each strategy is initiated with the same set of input parameters. All the output metrics are averaged over time and over the number of simulation repeats.

1. Input Parameters

The code has multiple input parameters that determine the performance of a specific strategy. Since no general underwater acoustic LAN network data are available, the data obtained from our experimental Seastar prototype, such as delays between transmissions, length of headers and size of utility packets, served in many cases as "general" input data. Often, the values are far from optimum but since this was the only way of calibrating the model versus a real system and since all strategies experience the same effects of these input parameters, a comparison between the four candidate strategies is still valid. Since many parameters have relatively similar effects on each strategy, a selection of critical parameters had to be made to emphasize the difference in character of each strategy. The following critical parameters are assessed to be useful and significant for expressing the fundamental differences in network strategy and the effects of varying these parameters are studied in the next section:

- Bit rate (high), used for transmitting data packets.
- Bit rate (low), used for transmitting utility packets and occasionally for retransmissions.
- Number of peripheral nodes.
- Packet size.
- Sub-packet size.
- Maximum number of poll or token retries.
- Maximum number of SRQ or ACK retries.
- Trigger level.

Each parameter is assigned a default value for the simulation as shown in Table 6. Most of these values are based on the settings of the Seastar prototype and the Teledyne Benthos ATM-885 modems as used during the in-water experiments. Although we expect performance improvements for a future version of Seastar compared to the prototype,

such as higher bit rates and shorter delays, we do not attempt to optimize the settings before all relevant parameters have been studied. Such optimization will be attempted for the case studies in Chapter VII.

PARAMETER	REF	UNITS	DEFAULT	PARAMETER	REF	UNITS	DEFAULT
Number of nodes	n	[ ]	6	packet size	D_p	[bytes]	2048
wake up time	t_{wu}	[s]	0.4	sub-packet size	D_{sp}	[bytes]	256
acquistition time	t_{acq}	[s]	0.28	bit rate (data)	R_{b1}	[bits/s]	800
size of utility packet	d_{ut}	[bytes]	9	bit rate (utility)	R_{b2}	[bits/s]	140
size of crc	d_{crc}	[bytes]	2	maximum SRQ retries	m_{srq}	[ ]	3
size of header	d_{nw}	[bytes]	14	maximum ACK retries	m_{ack}	[ ]	3
delay poll-data	t_{d1}	[s]	1	maximum poll retries	m_{poll}	[ ]	3
delay data-poll	t_{d2}	[s]	2.9	maximum token retries	m_{token}	[ ]	3
delay manual	t_{d3}	[s]	0	trigger level 0=min 1=max	α	[ ]	0.05
delay data-SRQ	t_{d4}	[s]	0.7	simulation period	T	[hrs]	10
time out period	t_{d5}	[s]	7.5	simulation repeats	A	[ ]	100

Table 6 Input parameters, including abbreviations used for reference. Most of the default values are based on the observed performance of the Seastar prototype during the in-water experiment.

2. Functions and Threshold Levels

The simulation code uses functions to perform certain calculations. For example, function INI.m is used to set network parameters, and function COLLECTDATA.m is responsible for collecting the performance data of the networks initialized with these parameters. Each network strategy is implemented as described in Chapter IV by a function containing multiple loops to simulate an event-driven network operation of finite duration. The first strategy function that was inserted in the simulation, however, represented an almost exact copy of the Seastar prototype and was used to test and calibrate the several other functions so that the output values matched the performance of both the in-air as well as the in-water experiments.

Each strategy function calls event functions to perform specific actions such as polling, data transmission or SRQ. Many of these functions contain pseudo-random number generators that are summoned each time the function is called. The numbers that are generated are compared to threshold levels that are set by a single parameter in the INI.m function, known as the threshold α. This is done for convenience, and to ensure a

consistent relation between these levels. The levels that are set by α are called L1, L2, L3, L23, L4, and L5 and determine the probability of events to occur, as can be seen in Figure 46. If a random number is generated between L1 and L2, it is interpreted as an unsuccessful poll or token and so a data packet is not transmitted from the peripheral node. Depending on the strategy, a retransmission will follow until the maximum number of retransmission attempts is reached. If a poll or token is successfully received by a peripheral node, a new random number is generated. If the value of this number lies between L2 and L3, the data packet is considered corrupted and the transmission is unsuccessful. The level L23 determines if the failure involves the header of the packet or one or more sub-packets. In the case of a corrupted header, a time-out period is activated, followed by a full retransmission if the network strategy permits. In the case of one or more corrupted sub-packets, a retransmission follows, again depending on the strategy, until the full packet is successfully received or until the maximum number of retransmissions has been made. The success of a retransmission is determined by L4. The level L5 just sets the lower level to zero.

These sequences of events, including retransmissions and dropped packets, then have an impact on the RTT and/or the amount of data transmitted. Flags can be set by functions to memorize actions that require follow up during the subsequent cycle, such as retransmissions in case of T3A or the occurrence of a packet-out-of-sequence situation.

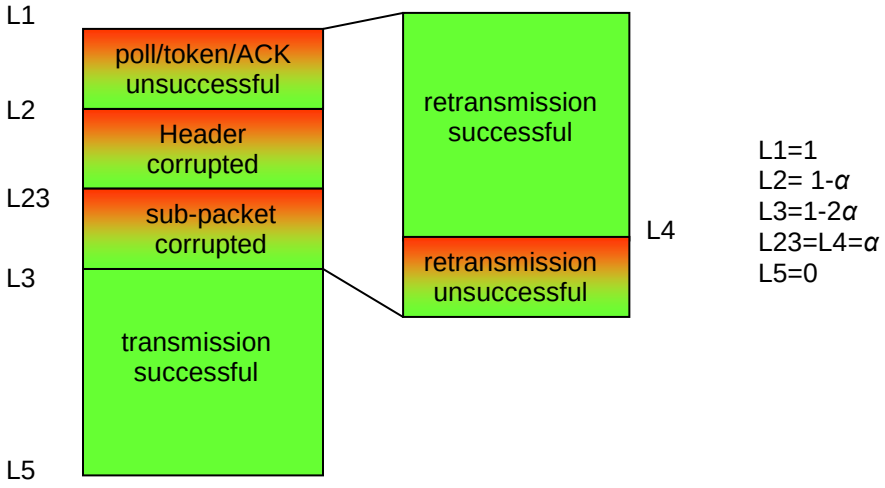

Figure 46 Graphical representation (not to scale) of the organization of levels as set by threshold α. The levels determine the probability of a certain event to happen.

Because of the random occurrences of events and the large variations in output that this process generates, our performance analysis depends on a Monte Carlo-type method to find statistically significant output values. Although this method, as introduced in [42], describes a statistical approach to study differential equations that occur in various branches of the natural sciences and since we do not provide these differential equations, the method is applicable to our case since it finds the most likely outcome of a multi-parameter process. By having the simulation produce a multitude of randomly generated possible outcomes and average them, we find a most likely behavior of a specific strategy under certain conditions. The Monte Carlo method requires a significant number of outcomes, hence the multiple repetitions of a network simulation over a significantly long time. As Table 6 shows, all simulations were conducted for a simulated time of 10 hours and averaged over 100 realizations. To illustrate this numerically, we consider an average performance outcome (e.g., latency=160 s) for the four strategies using the default settings of Table 6 in a noise-free environment (α=0). During these 100 realizations of 10-hour network operations, approximately 135,000 transmissions occurred and were used in calculations to estimate a most probable outcome. This count varies, of course, with changing parameters (e.g., longer packets reduce the amount of transmissions, higher bit rates increase this amount) but it shows that the number of events is sufficiently large for applying a Monte Carlo-like approach. We will now discuss how the output parameters are obtained.

3. Output Metrics

Channel utilization, information throughput, latency, and dropped packets are measured as follows. Each address receives a poll or token utility packet and transmits its data. The total time T_{xmit}, including overhead and retransmissions required for performing this round-trip transmission and the total amount of information *DATA* including retransmitted packets, is stored for each address. The simulation also keeps track of the number of dropped and successful packets. These variables are used to update the total transmission time and total transmitted information for each node (T_{modem} and D_{modem}, respectively), for each cycle (T_{cycle} and D_{cycle}, respectively) and for the total simulation

duration (T_{total} and D_{total}, respectively). The latency is calculated by summing T_{cycle} for each cycle. The information throughput over the period *T* is calculated by dividing D_{total} by T_{total}. The channel utilization over the period *T* is obtained by taking the ratio of the information throughput (D_{total}/T_{total}) and the bit rate R_{b1}. The percentage of dropped packets during the period *T* is simply the ratio of the dropped packets over the total number of transmitted packets. Finally, all the calculated values are averaged over the total number of realizations *A*, resulting in a set of statistically significant output parameters for channel utilization, information throughput, latency, and dropped packets.

4. Limitations

The simulation has certain limitations in representing the performance of the candidate network strategies. First of all, the strategies themselves are ideal models of possible future implementations. Packets ending up out of sequence, for example, can easily be avoided in this simulation since corrupted packets will either be retransmitted or dropped. In order to analyze actions responding to the detection of a packet that is out of sequence, a packet that is dropped after an unsuccessful poll or token is artificially placed out of sequence. The network now has to solve this situation during its next cycle. In a real situation this packet would just be dropped or retransmitted depending on the network design.

Although the simulation has been calibrated using experimental data in terms of performance metrics, the threshold α is just a value between 0 and 1 and the levels that depend on α do not represent true noise levels or SNR values. The simulation can therefore not be used to analyze the performance in a specific geographic region or to determine network settings prior to operational deployment. This would require a more sophisticated model and additional environmental input parameters.

Another important limitation is that the propagation delays are set for a fixed inter-nodal range of 500 m and therefore do not provide the flexibility to analyze network

performance when repositioning the nodes. Nor does it support analysis of mobile nodes. Generality of the simulation was not possible due to time constraints, but should be relatively easy to implement.

Overall, the simulation is found useful for providing performance comparisons of the four network strategies of interest. The code is flexible enough to analyze other forms of networks strategiesof so required. We now proceed by using the code for a parametric analysis of the strategies P1D, P1E, T2B, and T3A in terms of channel utilization, information throughput, latency, and dropped packets.

B. PARAMETRIC ANALYSIS

Throughout this section, we vary one relevant parameter at a time, while keeping all others default as given by Table 6. In some instances it is necessary to study the effects of a certain parameter in more depth, which may require adjusting other parameters. Deviations from the default settings will be clearly announced. Setting parameters to values that have earlier in this thesis shown not yet realistic is justified in anticipation of future technical improvements. Another justification is that trends become more clear and differences more profound when analyzing over a larger range.

1. Bit Rate

Recall that the asymmetric concept involves two bit rates, a high bit rate for data transfer R_{b1} and a low bit rate for utility packets R_{b2}. The simulation for analysis of R_{b1} is conducted for $R_{b1} = [500, 1000, 4000, 10000, 20000, 40000]$ bits/s. Increasing R_{b1} over this range, as is done in Figure 47, which shows the effect on utilization, information throughput, latency and dropped packets, does not result in a linear improvement of the information throughput and results in very low channel utilization for all network strategies. At large values for R_{b1}, the network performance is limited by the amount of overhead, consisting of (propagation) delays and utility packets. This also sets a lower limit for the latency.

To study the influence of communications overhead in more depth we consider the following input parameters: $t_{wu} = 0.2$ s, $t_{acq} = 0.14$ s, $t_{d1} = 0.7$ s, $t_{d2} = 0.7$ s, $t_{d5} = 3.5$ s and $R_{b2} = 4000$ bits/s. With these improved values, a better information throughput is observed (see Figure 48 for P1D) but overhead remains a constraint.

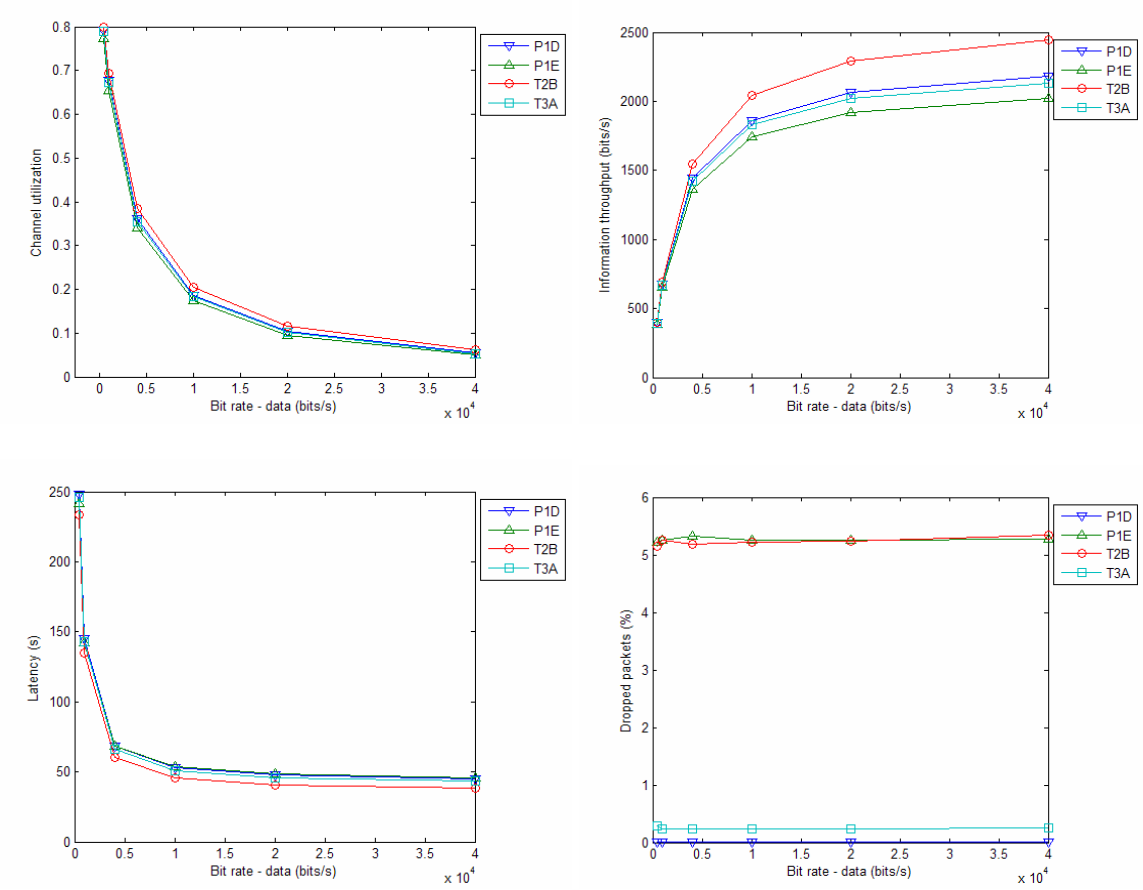

Figure 47 Increasing the bit rate R_{b1} (horizontal axis), expressed in terms of utilization, throughput, latency and dropped packets, has a limited effect on improving the network performance because of the increasing relative influence of overhead. All other input parameters other than R_{b1} are set to default values.

It may be obvious that P1D proved to be the most reliable network type under these conditions because of its SRQ ability. The number of dropped packets in the simulation is independent of the bit rate. Based on experiments described in Chapter IV it should be taken into account that higher bit rates, when caused by a reduction of

redundant bits, generally do result in an increase in transmission failures. T2B demonstrates the best information throughput and lowest latency but at the cost of dropping 5–6% of the packets. Overall, all network strategies were affected to a similar extent by an increased R_{b1} and performance for all strategies was limited by the correspondingly increased influence of overhead.

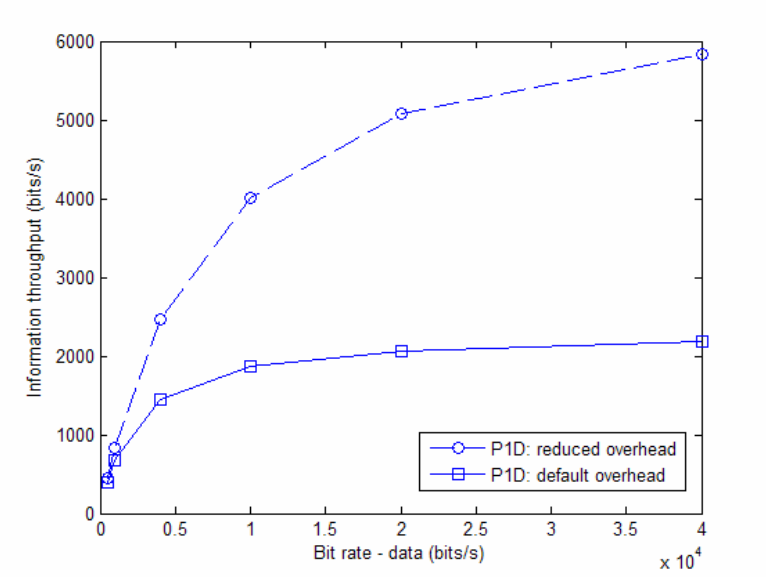

Figure 48 Reducing communications overhead, here shown for P1D, improves the information throughput of the network but still constrains perfromance at high R_{b1}. Input parameters for this study compare the default values with optimized values.

Increasing the utility packet bit rate by setting $R_{b2} = [50\ 100\ 200\ 400\ 800]$ improves the information throughput and reduces latency for all strategies. Strategy T3A especially benefits from a reduced overhead because time consumed by the additional hop is reduced. Notice that in Figure 49, unlike with R_{b1}, the channel utilization increases with increasing R_{b2}. Although increasing R_{b2} improves the network performance, we anticipate a relatively higher R_{b1} in future Seastar implementations and so it must be recognized that increasing the bit rate has its physical limits due to the inevitable overhead caused by delays and network headers.

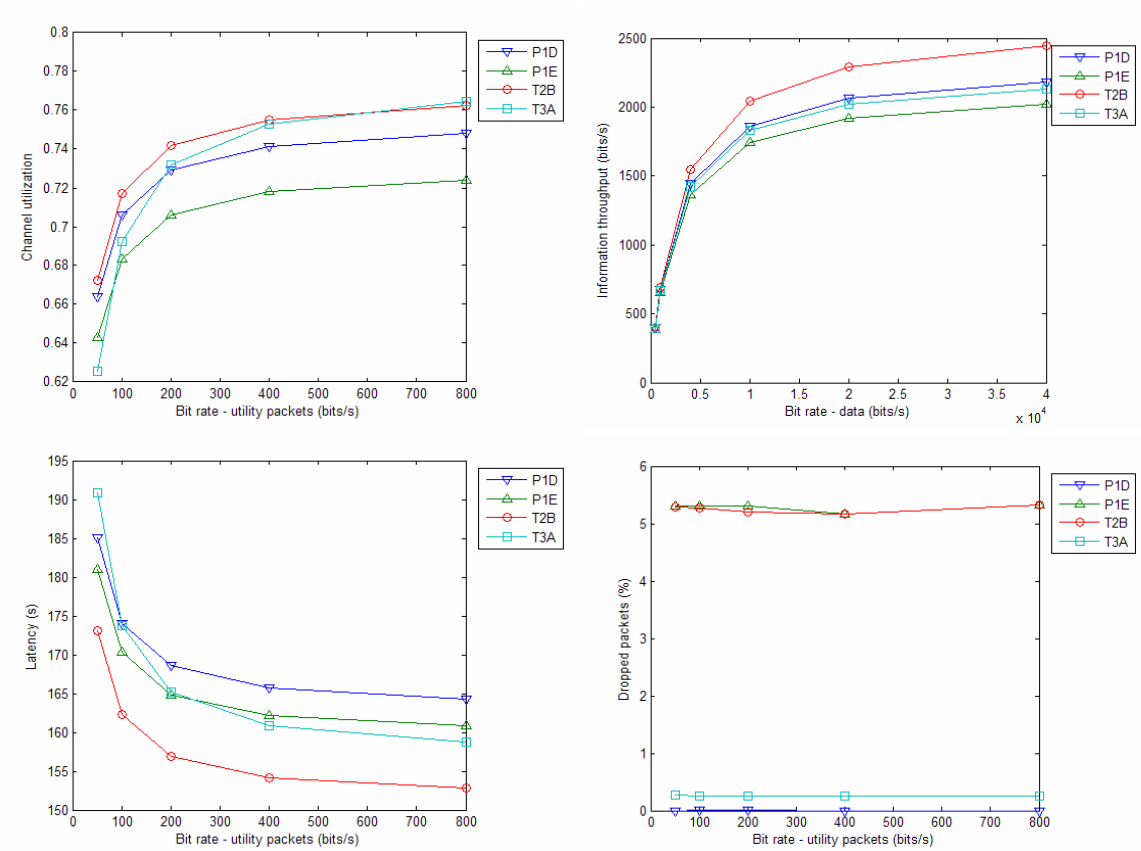

Figure 49 Increasing R_{b2} (horizontal axis) improves network performance and efficiency in terms of information throughput and channel utilization, and simultaneously reduces latency. All input parameters other than R_{b2} are set to default values.

2. Packet and Sub-packet Size

In general, larger information packets D_p result in good channel utilization and information throughput since the percentage of overhead is reduced. For an analysis where $D_p = [256, 512, 2048, 8192, 16348, 32768]$ bytes, it can be observed (see Figure 50) that the advantage in terms of information throughput and channel utilization of network type T2B is slightly reduced at packet size larger than 7 kilobytes (kbytes), although it still shows the lowest latency. P1D and T3A have about the same improved information throughput at larger packet sizes. Increasing packet size seems favorable but it unfortunately also results in a longer latency (see Figure 50).

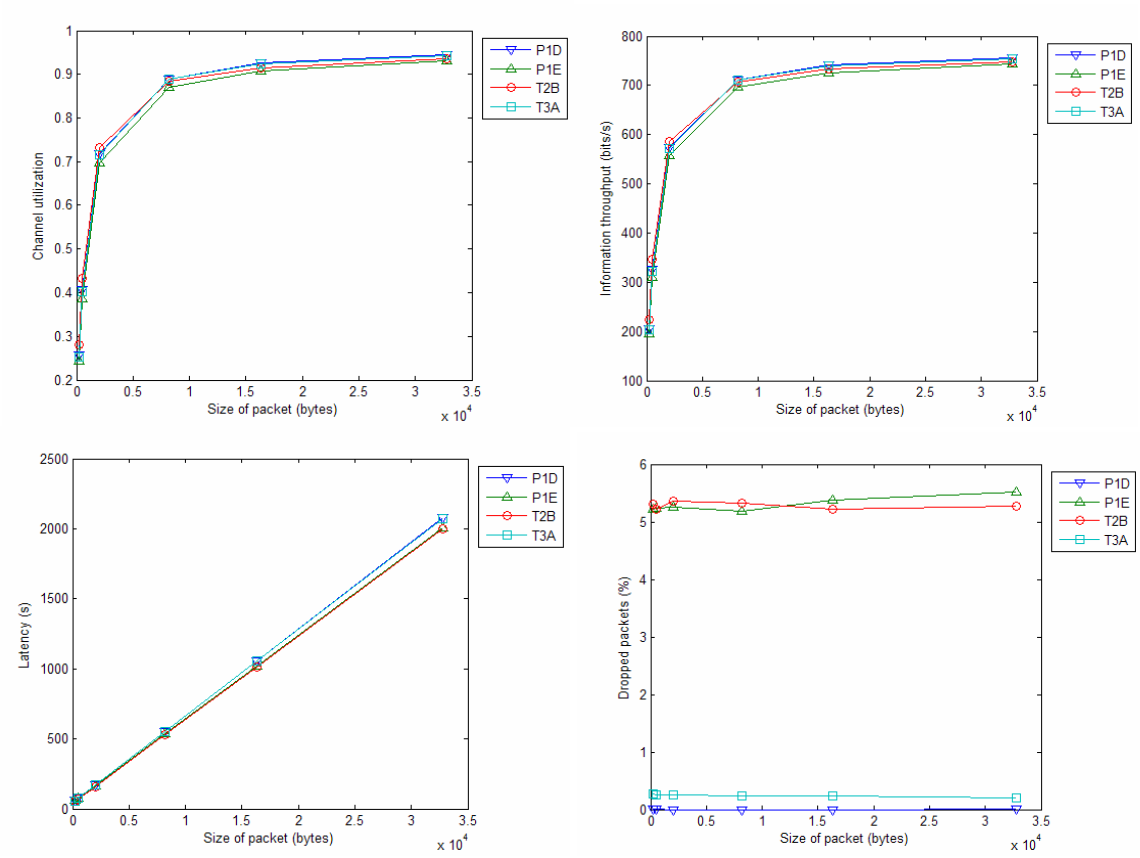

Figure 50 Increasing the size of packets D_p (horizontal axis) improves the information throughput and the channel utilization but has a negative effect on the latency. All input parameters other than D_p are set to default values.

The latency, however, can easily be reduced by increasing R_{b1}. Figure 51 shows the effect of packet size with increased R_{b1} (10000 bits/s) and R_{b2} (4000 bits/s) and setting $t_{wu} = 0.2$ s, $t_{acq} = 0.14$ s, $t_{d1} = 0.7$ s, $t_{d2} = 0.7$ s, $t_{d5} = 3.5$ s for P1D. Not only does this reduce latency, it also further improves the network performance in terms of information throughput. Increasing the packet size therefore needs to be considered in conjunction with other parameters but again, the negative effects of overhead are a limiting factor on information throughput.

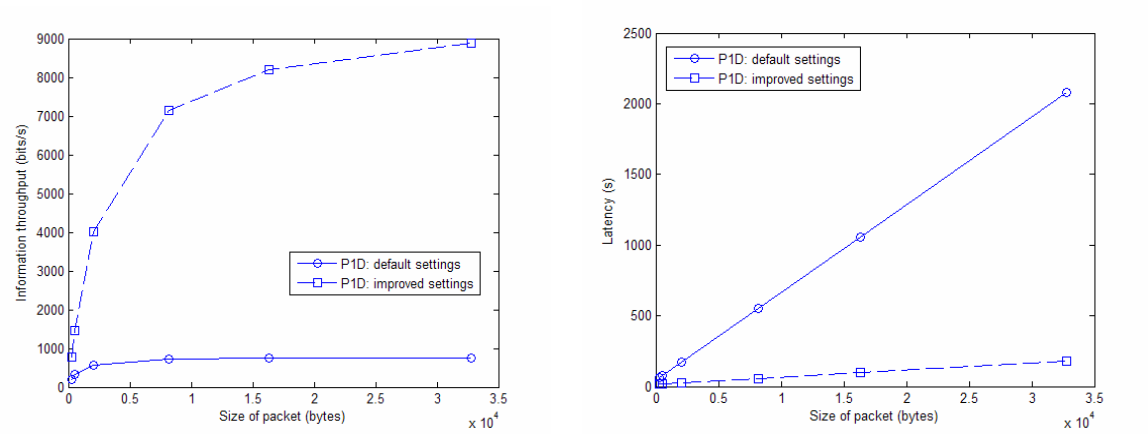

Figure 51 Simultaneously increasing packet size (horizontal axis) and bit rate (dashed line) improves both the information throughput (left) as well as the latency (right), as is shown here for P1D.

The percentage of dropped packets is unaffected by the packet size but a potential risk for longer latencies develops when noise levels increase and full packets need to be retransmitted, as is shown in Figure 52, where $\alpha = 0.2$ and the other parameters are set to default values.

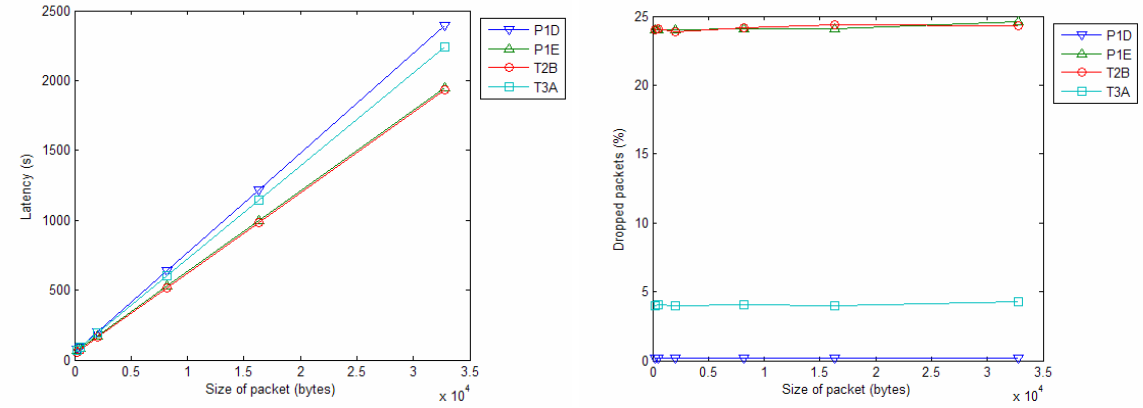

Figure 52 Increasing size of packets in a "noisier" environment ($\alpha = 0.2$). Full packet retransmissions cause long latencies for SRQ-able strategies. Non-SRQ strategies, on the other hand, drop an unacceptably large percentage of packets. All input parameters other than D_p and α, are set to default values.

We now turn our attention to the effect that the length of sub-packets has on the network performance and set $D_{sp} = [64,\ 128,\ 256,\ 512,\ 1024,\ 2048]$ bytes, with $D_p = 2048$ bytes. It should be mentioned that the D_{sp} parameter does not affect P1E and

T2B since these two strategies lack the provision to retransmit data packets. Varying the sub-packet size shows us some interesting results for network types P1D and T3A as shown in Figure 53.

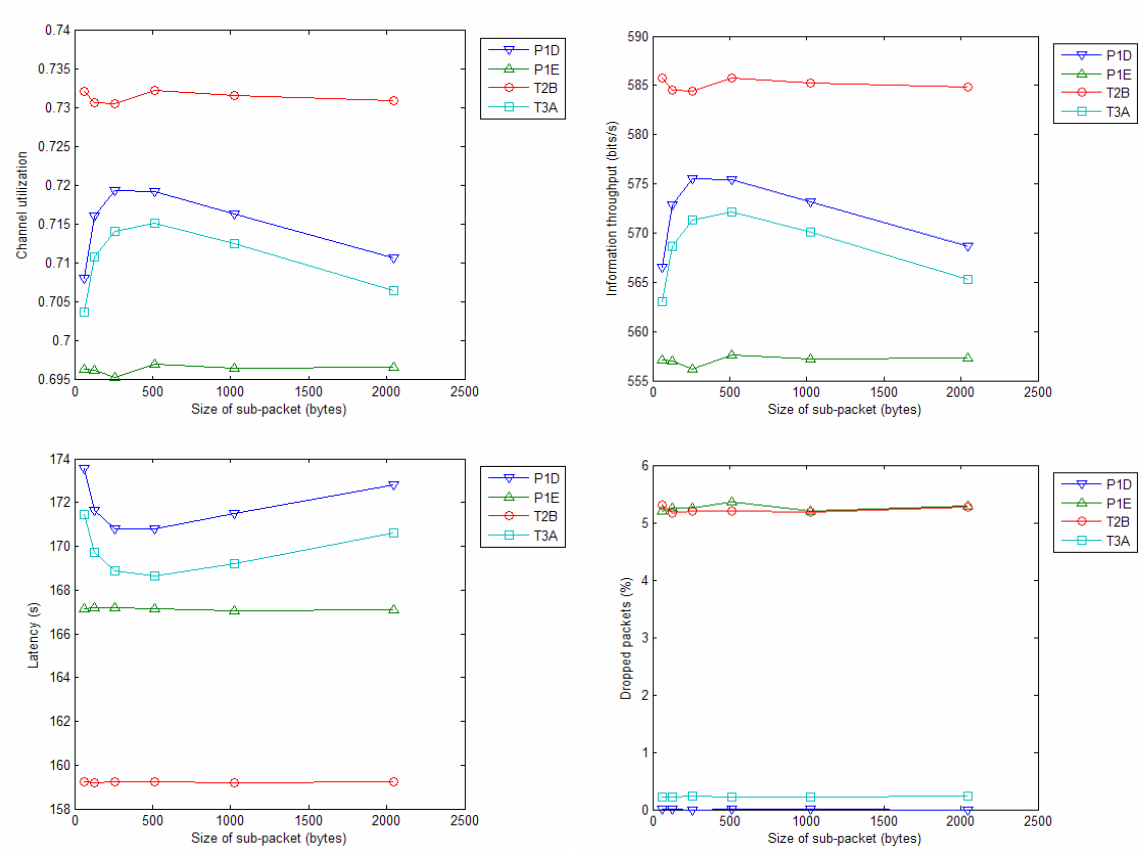

Figure 53 Reducing the size of sub-packets D_{sp} (horizontal axis) shows an optimum value near $D_{sp} \approx 500$ bytes under default conditions ($\alpha = 0.05$) for P1D and T3A. Since P1E and T2B do not use SRQ, changing D_{sp} does not have any effect. All input parameters other than D_{sp}, are set to default values.

Both show a maximum value for channel utilization and information throughput and a minimum value for latency, resulting in an optimum sub-packet size at $D_{sp} \approx 500$ bytes. At values of D_{sp} below the optimum, the network performance is degraded due to the additional CRC overhead associated with a larger number of sub-packets. At values of D_{sp} above the optimum, the network performance is degraded because of the lengthy retransmissions that need to be made. The position of the optimum is determined by the

amount of noise that is introduced. As an example, we set $\alpha = 0.2$ in Figure 54 to show that increased noise levels cause the optimum to shift to the smaller sub-packet sizes as may be expected.

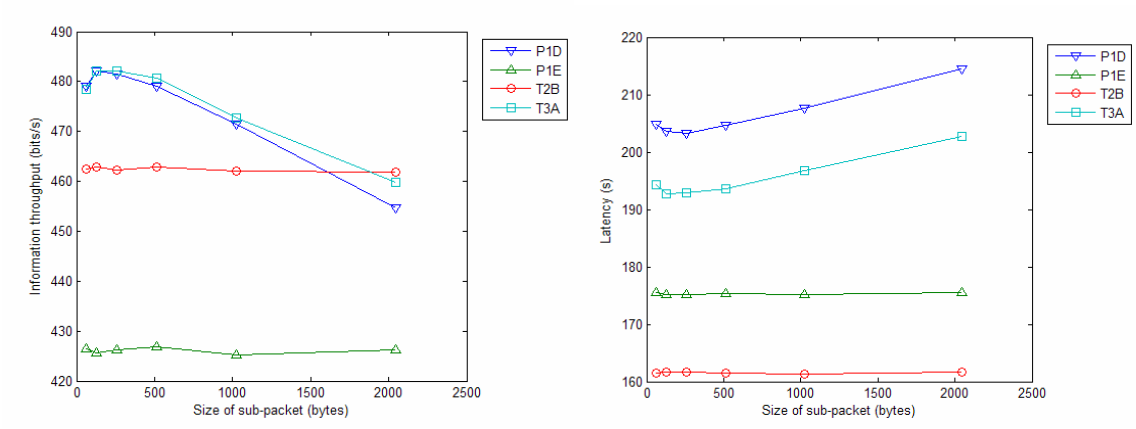

Figure 54 The position of the optimum size of sub-packets D_{sp} is determined by the amount of retransmissions that is required. Increasing the "noise" by setting $\alpha = 0.2$ causes the optimal D_{sp} to shift to smaller values, in this case $D_{sp} \approx 100$ bytes. All input parameters other than D_{sp} and α, are set to default values.

In general, large sub-packets will cause longer delays in a noisy environment since many retransmissions are anticipated. Breaking up the packet into many small sub-packets reduces the latency. In relatively noise-free environments, however, the negative influence of CRC overhead due to the larger number of sub-packets contributes to a decreased information throughput as well as longer latency.

Increasing bit rates and reducing delays ($R_{b1} = 10000$ bits/s, $R_{b2} = 4000$ bits/s, $t_{wu} = 0.2$ s, $t_{acq} = 0.14$ s, $t_{d1} = 0.7$ s, $t_{d2} = 0.7$ s, $t_{d5} = 3.5$ s and $\alpha = 0.2$) removes the presence of an optimum D_{sp} and favors T2B and T3A over P1D and P1E as is shown in Figure 55.

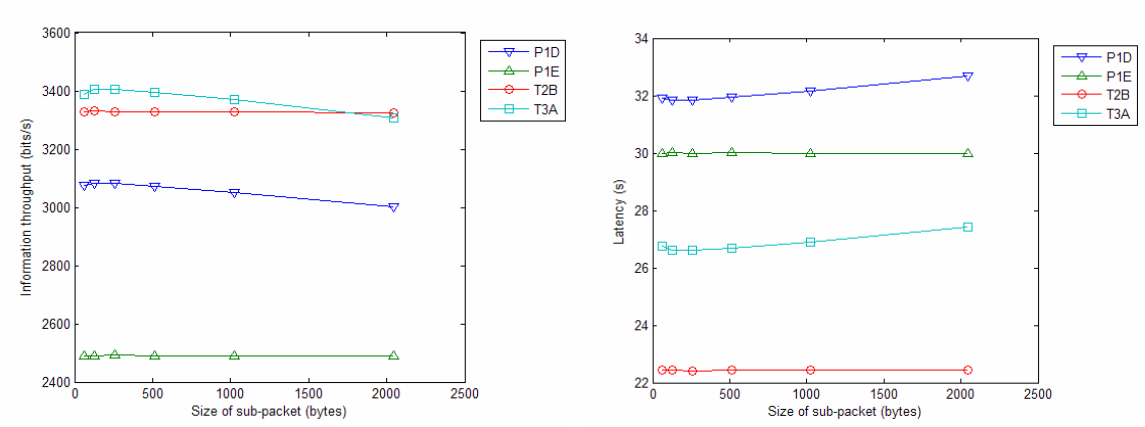

Figure 55 Operating at higher bit rates with reduced delays as described above reduces the appearance of an optimum value for D_{sp} and favors T2B and T3A over P1D and P1E.

In general, small sub-packets are preferred in noisy environments and T3A is the preferred strategy under these conditions.

3. Number of Peripheral Nodes

Recall that our simulation uses fixed propagation delays whereas a change in the number of nodes actually requires adjusting these ranges. For example, from Table 2 we find a range reduction of 146 m for $n = 4$ which agrees with 0.097 s for a sound speed of 1500 m/s. Even when considering a round trip transmission of 10 s, the difference in transmission time is less than 1%, so ignoring the range adjustment has a negligible effect on the calculations. As expected, the number of nodes only has an effect on latency. Figure 56 shows that, except for T3A, there is no impact on channel utilization, information throughput or dropped packets when analyzing for $n = [4,\ 6,\ 8,\ 10,\ 12]$ peripheral nodes.

Using default settings, T3A shows a degraded performance in terms of utilization and information throughput with increasing number of modems.

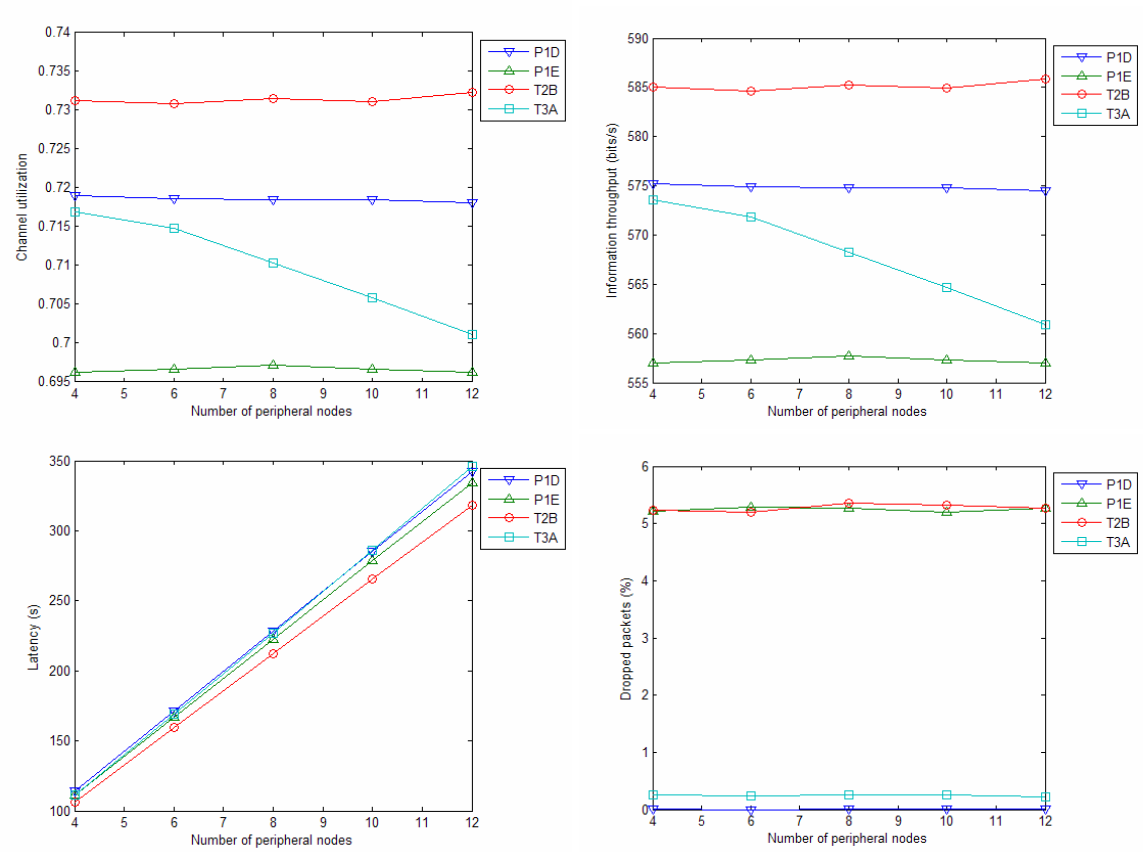

Figure 56 For P1D, P1E and T2B, the number of peripheral nodes *n* (horizontal axis) only affects the latency of the network. The information throughput of T3A decreases with increasing *n* because of the increasing length of the token. All input parameters other than *n*, are set to default values.

A surprising result arises when the low bit rate R_{b2}, which determines the transmission speed of utility packets such as the token, is increased from the initial value of 140 bits/s to higher speeds. For this analysis we set $R_{b1} = 10$ kbits/s. Figure 57 shows the appearance of a maximum value for information throughput, shifting to the right (larger number of modems) for increasing R_{b2} and finally resulting in the reverse effect, namely an improved performance for increasing number of nodes. Recall that T3A requires a hop through the central node to update the token and that each additional node adds bytes to the token. The explanation for the phenomenon observed in Figure 57 can

be found in the fact that at low R_{b2} (e.g., 140 bits/s) the performance of T3A is dominated by the overhead due to the increasing length of the token. At high R_{b2}, the reduced impact of the "central node hop" through the addition of more peripheral nodes dominates the relative loss of an increased token length. Careful design of the token as well as applied bit rates for network type T3A is therefore paramount.

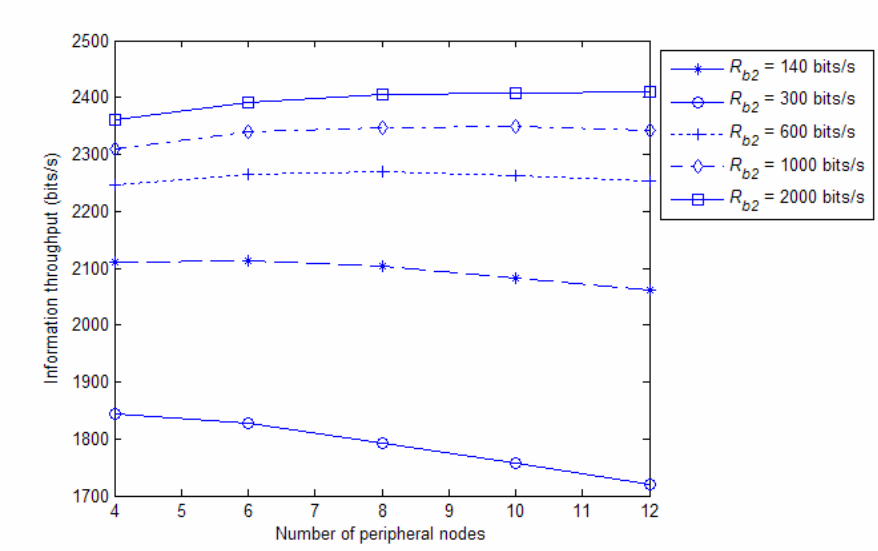

Figure 57 Increasing the number of peripheral nodes (horizontal axis) at $R_{b2} = 140$ bits/s ($R_{b1} = 10$ kbits/s) results in a decreased information throughput for T3A. Increasing R_{b2} as shown to $R_{b2} =$ [300, 600, 1000, 2000], reverses this effect.

In general it can be stated that more nodes mean a higher sensor/modem density at the cost of increased latency.

4. Number of Retransmissions

The maximum number of retransmissions m determines how many times a specific packet is allowed to be retransmitted before it is dropped. For this analysis we set $m = m_{srq} = m_{ack} = m_{poll} = m_{token} = [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 8]$. The effect of increasing this parameter clearly depends on the amount of noise imposed upon the network and in order to observe a sufficient number of retransmissions and emphasize the differences, we set $\alpha = 0.2$. Since P1E and T2B do not allow for retransmitting corrupted data or utility packets, the maximum number of retransmissions does not affect these strategies. The

outcome of this analysis (see Figure 61), should be interpreted with some reserve. Recall that, for simulation purposes, a packet is put out of sequence once the maximum number of poll or token retransmissions has been achieved. Usually, an event like this does not happen very often but since the simulation for this specific parameter includes extremities (e.g., $\alpha = 0.2$ in combination with small values for *m*), packets do end up out of sequence frequently. When that happens, full retransmission occurs at a bit rate of R_{b2} instead of R_{b1}, which has a dramatic effect on the network performance of P1D and T3A. Although the results for low *m* may underestimate the performance, the analysis is useful because it shows the advantage of being able to retransmit corrupted packets. At the same time it also shows that the performance levels off at $n > 5$.

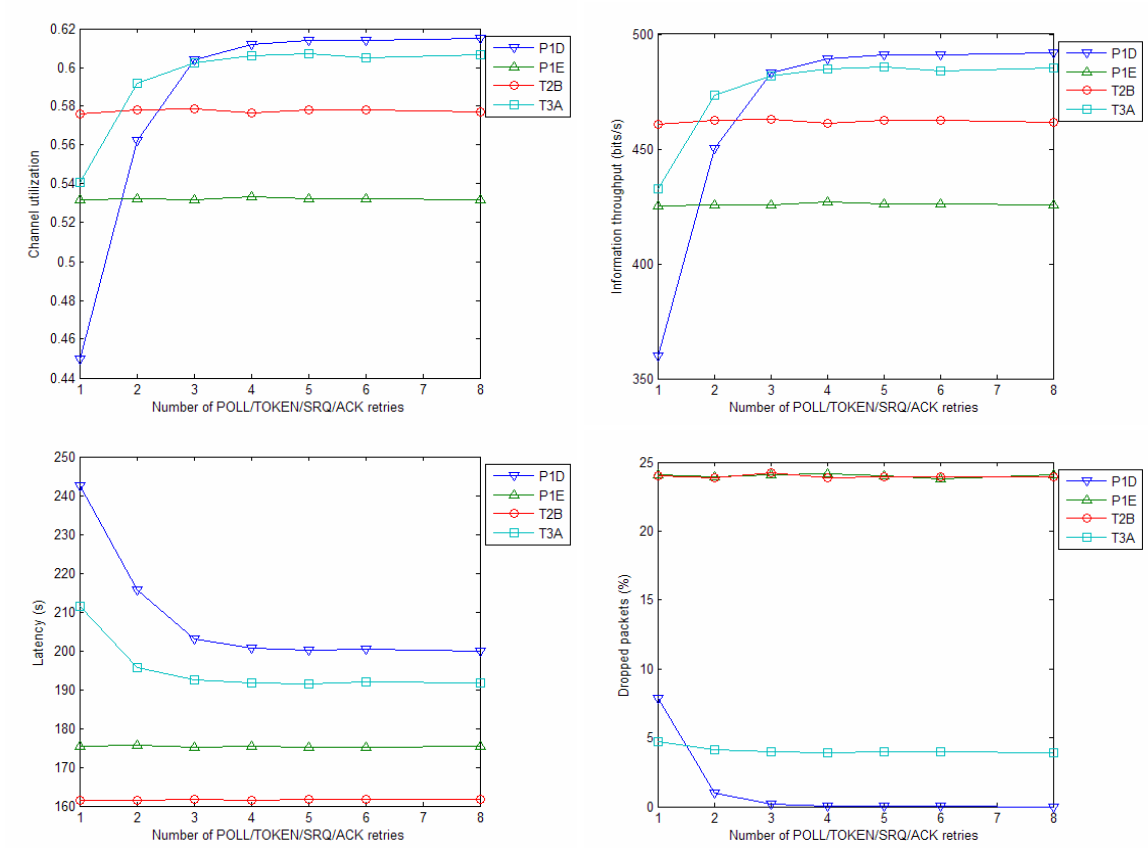

Figure 58 Effect of number of retransmission retries *m* (horizontal axis) for "very noisy" conditions ($\alpha = 0.2$). The ability to retransmit packets ensures a relatively good information throughput at a low dropped packet percentage, at the cost of increased latency. All input parameters other than *m* and α, are set to default values.

In general, network strategies that are capable of retransmitting packets ensure reliable data transfer in terms of dropped packets and information throughput. For "noisy" conditions, the cost of longer latency for P1D and T3A is acceptable, certainly when considering the large number of dropped packets that arise from P1E and T2B.

5. Noise

The introduction of the effects of "noise," represented qualitatively by the threshold α, has, in some cases, already been discussed in combination with previous parameters. In the figures, α is expressed as a percentage and can be interpreted as a qualitative variable that determines the success rate of a transmission. The threshold does not directly refer to *SNR*, *SL*, *NL* or *TL*, but can be used as a "knob" to set system or channel degradation. As an example, $\alpha = 0.1$ means that 10% of packets of any type are initially unsuccessful, and the experienced α during the in-water experiments was 0.01.

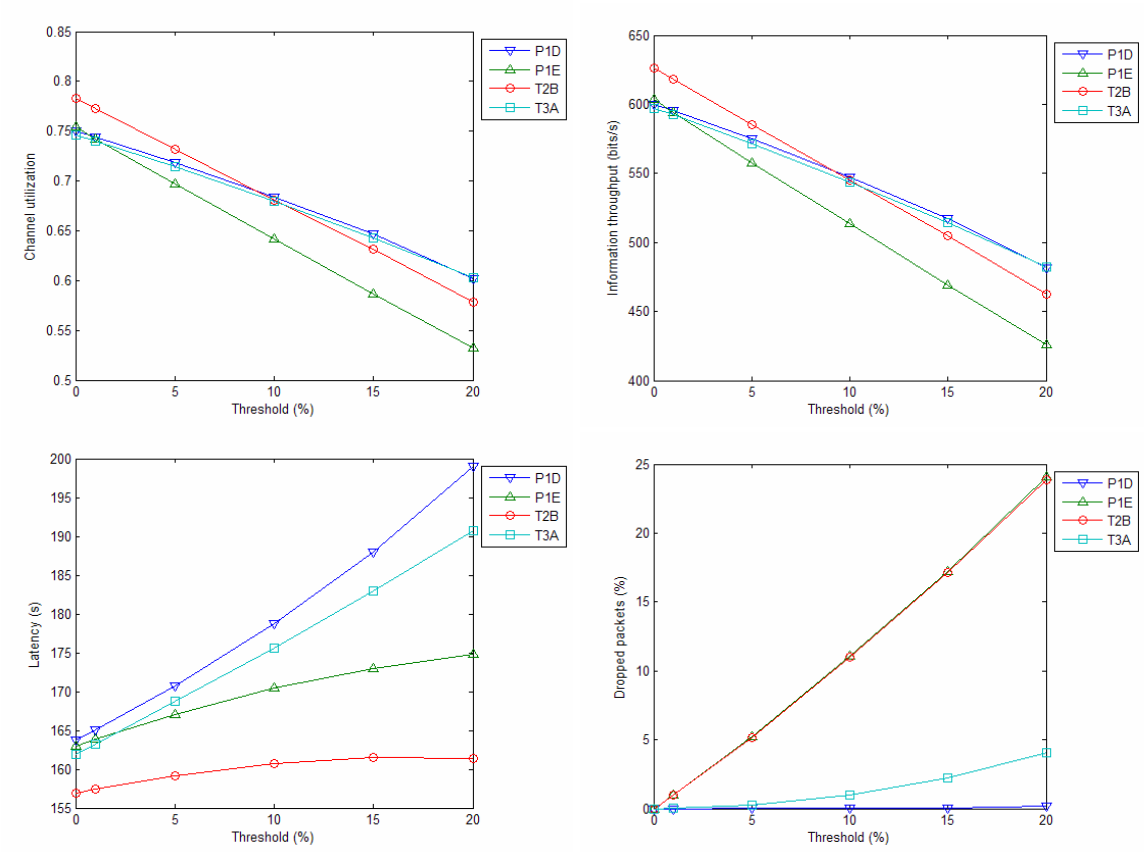

Figure 59 The effect of "noise" on the output metrics is set by the threshold (horizontal axis). All input parameters other than α, are set to default values.

For $\alpha = [0,\ 0.01,\ 0.05,\ 0.1,\ 0.15,\ 0.2]$ the simulation puts P1D and T3A in favor over P1E and T2B in terms of information throughput when $\alpha > 0.1$. This value for α shifts up when setting $R_{b1} = 10000$ bits/s, $R_{b2} = 4000$ bits/s, $t_{wu} = 0.2$ s, $t_{acq} = 0.14$ s, $t_{d1} = 0.7$ s, $t_{d2} = 0.7$ s, $t_{d5} = 3.5$ s, as can be seen in Figure 60 but the trend remains the same.

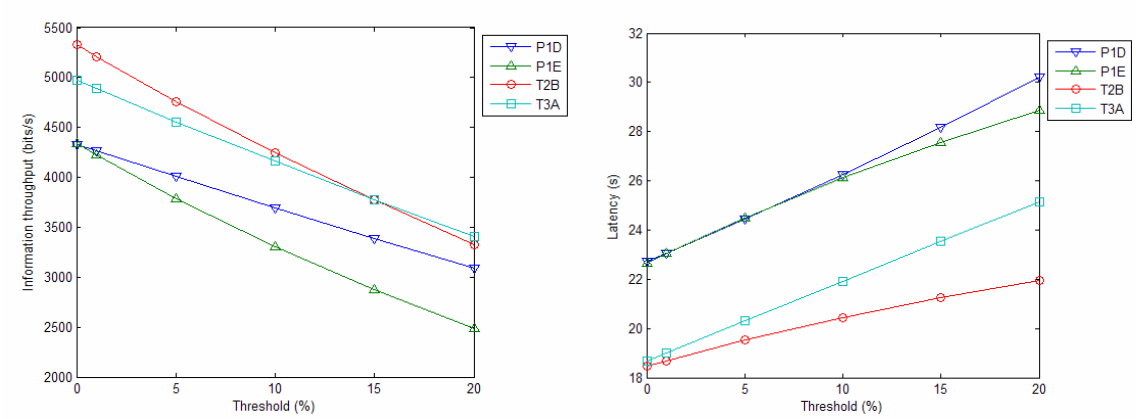

Figure 60 Parametric analysis for α or packet error rate (horizontal axis) at higher bit rates and reduced delays as described above. The strategies shift relative from each other but still show a similar trend as with the default settings.

The best metric to analyze the performance of network strategies under the influence of noise is, however, not always information throughput. In case of $\alpha > 0.05$, the dominant factor for choosing a strategy will almost certainly be the amount of dropped packets, which is unacceptably high for P1E and T2B. Even when the maximum number of allowed retransmissions for P1D and T3A is reduced to $m = 1$ (Figure 61), P1E and T2B show a relatively very poor performance in terms of dropped packets. Referring again to the in-water experiments that were done with the prototype, where $\alpha = 0.01$, we state that T2B would perform well enough under these low-noise conditions, but over the full range, when reliable message delivery is required, T3A performs best, followed by P1D.

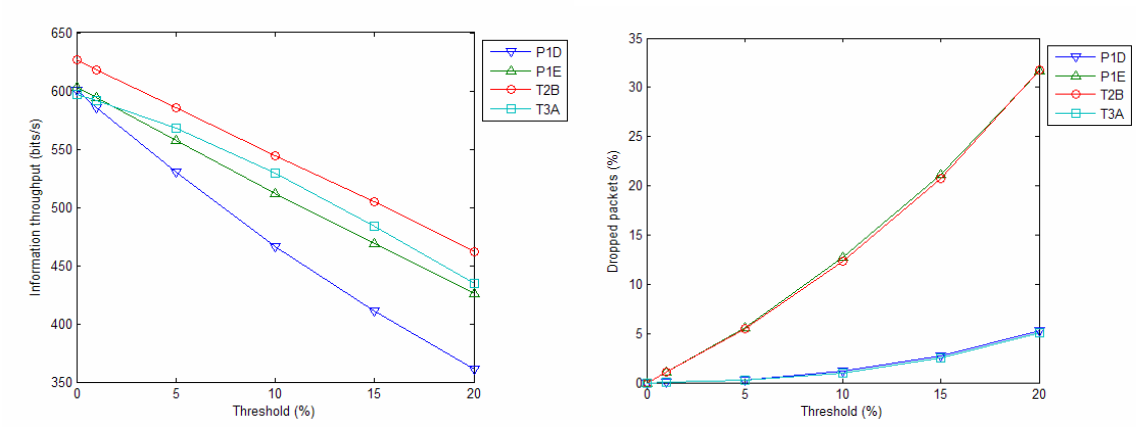

Figure 61 Even though P1D and T3A are only allowed one retransmissions ($m = 1$), they remain the preferred strategy under "noisy" conditions (horizontal axis) when considering the percentage of dropped packets.

C. TRADEOFFS

It should be clear by now that *the* optimum strategy does not exist. Defining an optimum strategy depends on operational requiremens, such as required reliability, latency and throughput. We have also seen that channel noise plays an important role in determining the strategy. In order for the reader to comprehend the results of the parametric analysis in a nutshell we try to summarize the generally observed trends in two ways.

First, we summarize the results of the parametric analysis graphically (see Figure 62), and indicate the effect that increasing or reducing the value of a certain parameter has on channel utilization, information throughput, latency and dropped packets. As an example, increasing the data bit rate R_{b1} generally has a negative (red) effect on channel utilization (upper left field) but a positive (green) effect on information throughput. It must be emphasized that Figure 62 shows *general* effects and that a specific strategy or changing certain parameters may influence or change to what extent a certain effect is observable. Also note that an additional parameter t_d is expressed explicitly to clarify that the parametric analysis includes effects of reducing delays as part of the analysis when overhead is reduced.

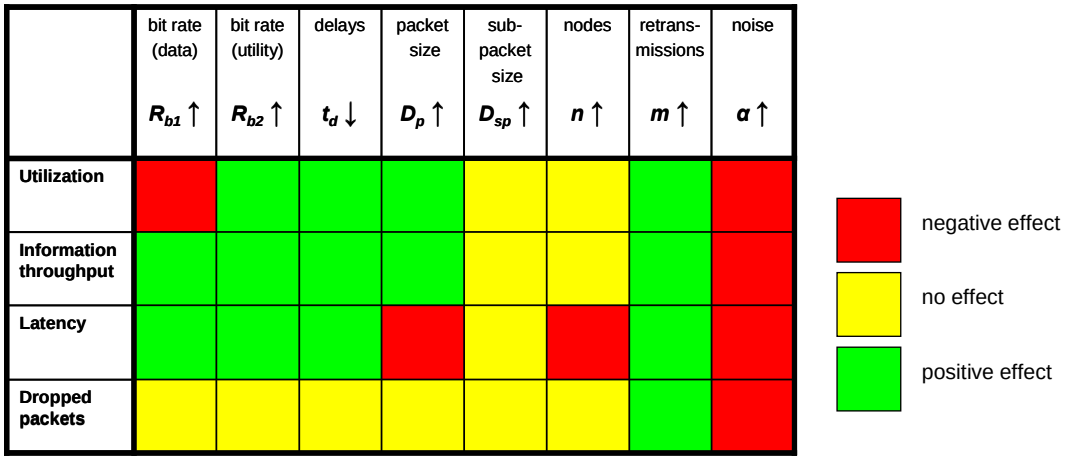

Figure 62 Summary of parametric analysis. Columns indicate increasing (arrow up) or decreasing (arrow down) parameter values and the effect of this on the metrics used (rows).

Noise is the only parameter that generally cannot be influenced and that has a profound effect on the preferred network strategy. To express the influence of noise on network performance more specifically, we dedicate a second summary to the relationship between the performance of a specific network strategy and the various parameters, under low-noise ($\alpha = 1\%$) and high-noise ($\alpha = 20\%$) conditions, respectively.

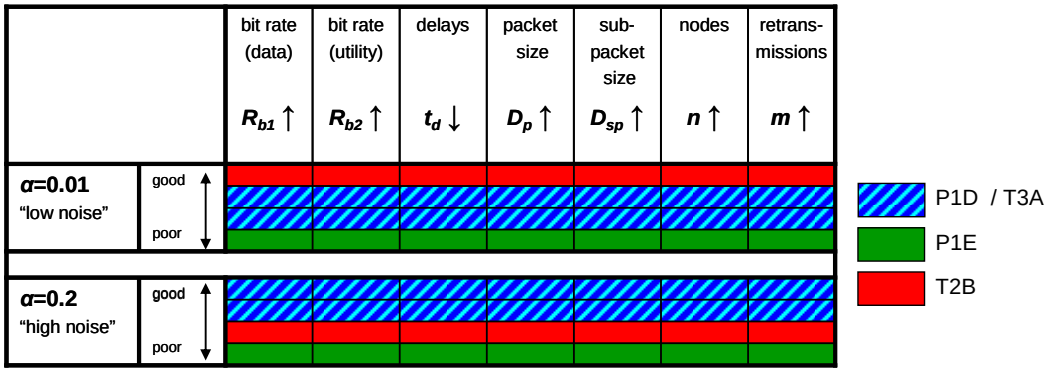

Figure 63 Performance of network strategies for $\alpha = 0.01$ (upper table) and 0.2 (lower table), respectively, for designated parameters. T2B excels in very low-noise environments. In noisy environments, P1D performs best in terms of reliability whereas T3A performs best in terms of information throughput and latency.

Figure 63 reveals a couple of interesting trends, although one should interpret the figure with some reserve since the performances of some strategies are sometimes comparable (e.g., see Figure 50). Nevertheless, the following hard conclusions can be made.

Strategies P1E and T2B perform poorly under noisy (e.g., $\alpha > 0.02$) conditions because of the inability to retransmit packets. Strategy P1E also shows a relatively poor performance even under good conditions. In the case of both $\alpha = 0.01$ and $\alpha = 0.2$, P1D and T3A show almost always a comparable performance, which is good because it allows the choice between two different strategies depending on requirements. Strategy T3A performs slightly better in terms of channel utilization, information throughput and latency whereas P1D scores better in terms of reliability since it almost always maintains a zero dropped packet rate.

Considering all possible environmental conditions we therefore conclude the parametric analysis by stating that T2B shows superior performance in terms of utilization, information throughput and latency but is only useful if reliability in terms of dropped packets does not have the highest priority. Strategy P1E shows the poorest performance but may be preferred because it is independent of relative positioning between neighboring nodes. If reliability of data transfer is desirable (which it almost always is) then P1D and T3A both perform well. T3A is the fastest and indicates the highest throughput but is less flexible in terms of geometry because it depends on communication between neighboring nodes. Strategy P1D maintains the highest reliability because of its ability to retransmit multiple times but at the cost of longer latencies. We must not forget that hybrid or other forms of the above strategies are possible.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CASE STUDIES

In this chapter we use the simulation to consider possible future Seastar applications. The ideas for the case studies are generally inspired by the Underwater Persistent Surveillance (UPS) experiment that was conducted as part of the Monterey Bay 2006 field experiments [43].

A. ELECTROMAGNETIC RECONNAISSANCE

The first case study involves a fixed underwater reconnaissance system, consisting of magnetometers to detect magnetic anomalies associated with the passage of ships or submarines (e.g., see Figure 64). Upon detection, the magnetic signature of the target is determined and a rough tracking is obtained. The system is based on a Seaweb wide-area network where each Seaweb node serves as the central node for a Seastar LAN and each peripheral node includes a magnetic anomaly detector. We examine the network performance of a single Seastar LAN.

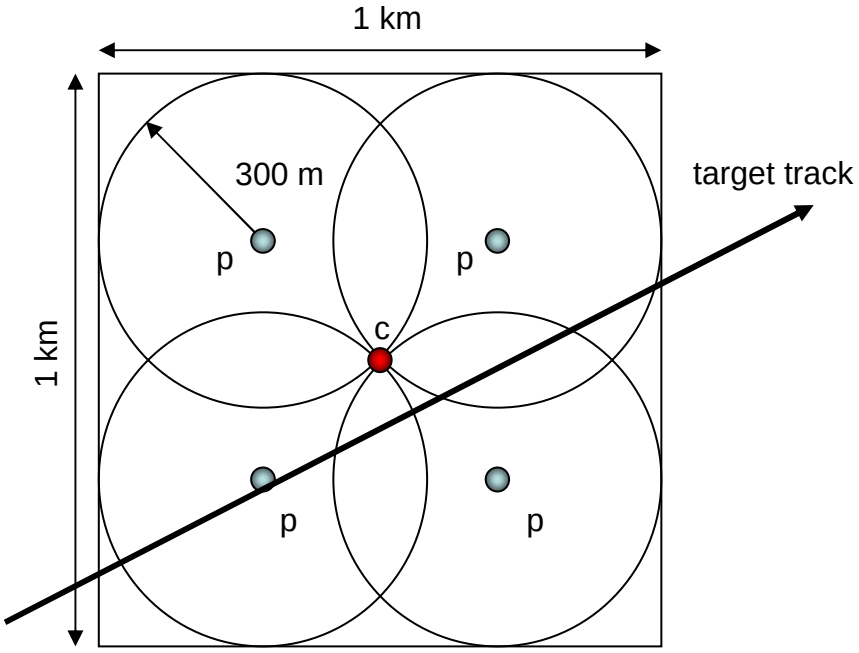

Figure 64 Schematic representation of a surveillance network with, in this case, four peripheral nodes with magnetic sensors having a 300-m detection range.

The operational requirements are as follows. The Seastar LAN should cover 1 km^2 and within this area, a target on a fixed heading should pass within detection range of at least two sensors to allow rough tracking. The magnetic sensor has a maximum detection range of 300 m. The target has a maximum target speed of 30 knots (≈ 55 km/hr). Near-real-time updates consisting of fused data from the central node is required with a latency of less than one minute. A raw data packet containing signature data that has been preprocessed by the magnetic sensor is assumed to have a size of 2 kbytes. Delivery of packets needs to be ensured, keeping in mind that the latency should not exceed 60 s. The network should operate under noisy channel conditions and disruption because the passage of loud targets should be expected.

The system default settings are optimized and are displayed in Table 7. Since the percentage of dropped packets needs to be minimized and the network is required to operate in noisy conditions, P1E and T2B are found not suitable and are not analyzed.

PARAMETER	REF	UNITS	DEFAULT
Number of modems	n	[ ]	var
wake up time	t_{wu}	[s]	0.1
acquistition time	t_{acq}	[s]	0.1
size of utility packet	d_{ut}	[bytes]	8
size of crc	d_{crc}	[bytes]	2
size of header	d_{nw}	[bytes]	12
delay poll-data	t_{d1}	[s]	0.4
delay data-poll	t_{d2}	[s]	0.4
delay manual	t_{d3}	[s]	0
delay data-SRQ	t_{d4}	[s]	0.4
time out period	t_{d5}	[s]	3

PARAMETER	REF	UNITS	DEFAULT
packet size	D_p	[bytes]	2000
sub-packet size	D_{sp}	[bytes]	var
bit rate (data)	R_{b1}	[bits/s]	var
bit rate (utility)	R_{b2}	[bits/s]	var
maximum SRQ retries	m_{srq}	[ ]	3
maximum ACK retries	m_{ack}	[ ]	3
maximum poll retries	m_{poll}	[ ]	3
maximum token retries	m_{token}	[ ]	3
trigger level 0=min 1=max	α	[ ]	var
simulation period	T	[hrs]	10
simulation repeats	A	[ ]	100

Table 7 Settings for reconnaissance case study. The value "var" refers to a variable that is an outcome of the simulation.

Our analysis first focuses on the required bit rate R_{b1} (see Figure 66). For $R_{b1} = [2000,\ 4000,\ 6000,\ 8000,\ 10000]$ bits/s, $R_{b2} = 2000$ bits/s, $n = 4$, $D_{sp} = 256$ bits, and $\alpha = 0.01$, we find that $R_{b1} > 2000$ bits/s keeps the latency well below the desired 60

s. Both P1D and T3A do not differ much from each other in performance and the small percentage of dropped packets (e.g., $\approx 0.01\%$) under these conditions for T3A is acceptable.

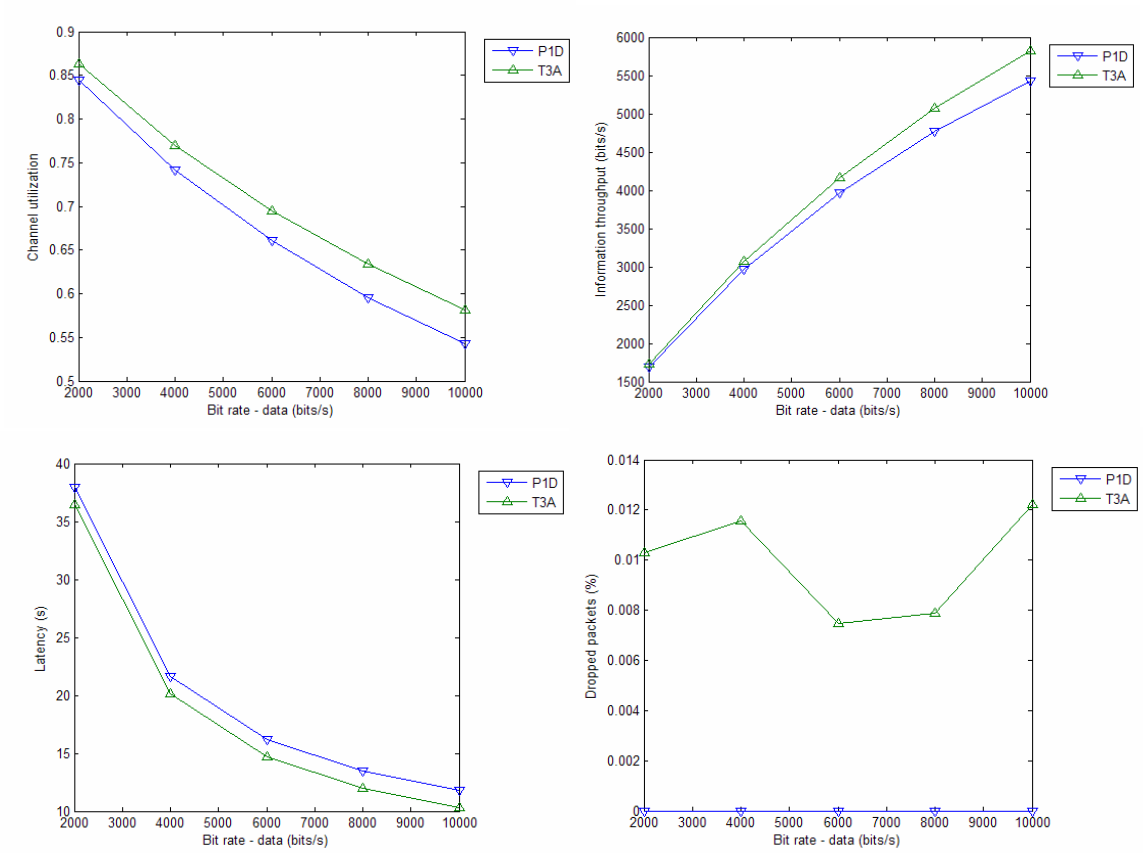

Figure 65 Effects of bit rate R_{b1} for case study A on the performance of a surveillance network as described above with $D_p = 2000$ bytes, $n = 4$, $R_{b2} = 2000$ bits/s, $D_{sp} = 256$ bits, $\alpha = 0.01$. For all R_{b1} shown, the latency is well below 60 s.

Increasing the packet size to $D_p = 10000$ bytes and keeping the settings as above (see Figure 66), requires a minimum $R_{b1} = 6000$ bits/s. This implies that MFSK modulation as described in Chapter III is not suitable for this kind of network within the available bandwidth, when the packet size is too large.

The previous chapter has shown that latency depends largely on the number of nodes. Since latency is a crititcal metric, it is necessary to analyze the impact that the number of peripheral nodes has on the latency for this case study. Based on the previous

bit rate analysis and MFSK-achievable bit rates, we set $R_{b1} = 3000$ bits/s, $R_{b2} = 2000$ bits/s, and let $n = [5,6,7,8,9,10]$ while keeping all other input parameter values as above. For these settings, the maximum number of peripheral nodes that is allowed by the latency requirement is limited to $n = 9$. The results are shown in Figure 67. Higher bit rates would allow more nodes for latencies less than 60 s; lower bit rates would limit the maximum number of nodes even more.

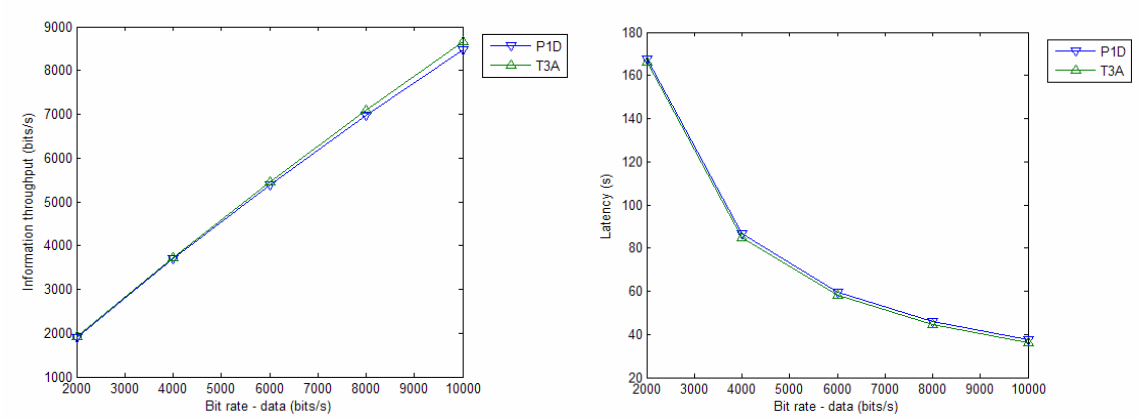

Figure 66 Effects of bit rate R_{b1} for case study A on the performance of a surveillance network as described above with $D_p = 10000$ bytes, $n = 4$, $R_{b2} = 2000$ bits/s, $D_{sp} = 256$ bits, $\alpha = 0.01$. For these settings, at least $R_{b1} = 6000$ bits/s is required to hold latency to 60 s.

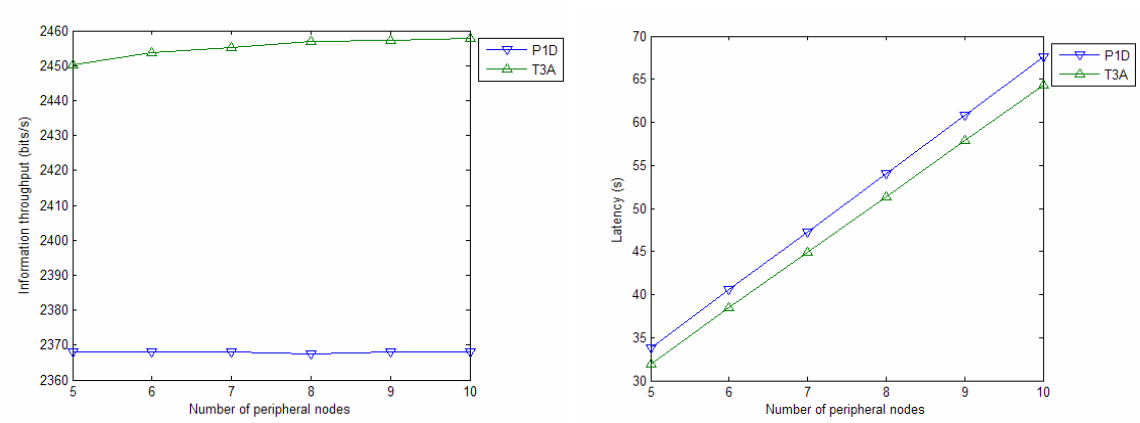

Figure 67 The number of nodes has a profound effect on latency and limits the maximum number of nodes *n* for case study A and $R_{b1} = 3000$ bits/s, $R_{b2} = 2000$ bits/s to $n = 9$.

We continue the analysis with the settings $n = 4$, $R_{b1} = 3000$ bits/s and $R_{b2} = 2000$ bits/s and now focus on the network performance under various "noise" conditions by letting $\alpha = [0, 0.01, 0.05, 0.1, 0.15, 0.2]$. It is clear from Figure 68 that P1D outperforms T3A in terms of reliability at a small additional cost in latency of about two seconds.

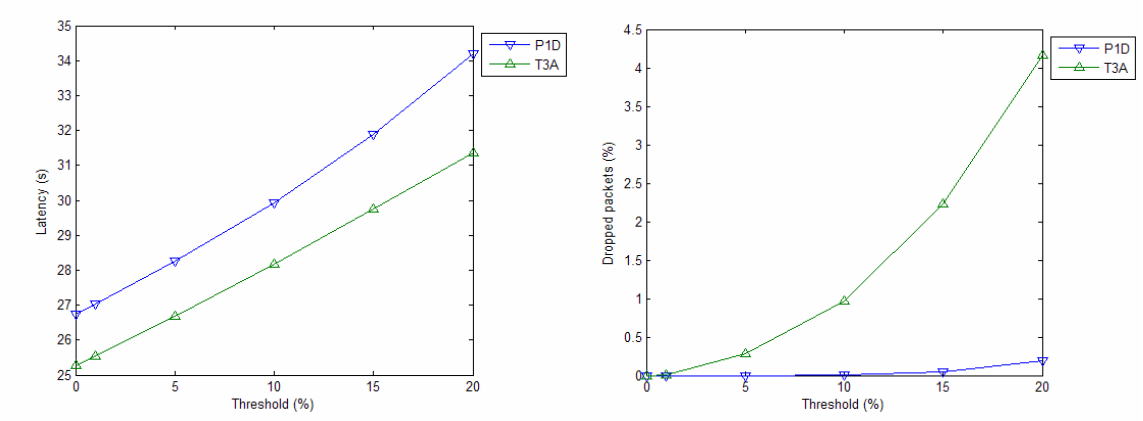

Figure 68 Noisy conditions for case study A with $n = 4$, $R_{b1} = 3000$ bits/s and $R_{b2} = 2000$ bits/s result in a degradation of T3A performance. The cost in latency that has to be paid to ensure packet delivery (P1D) is two seconds.

Considering that P1D is the most reliable and suitable network under all conditions, we conclude this study by determining a suitable number of sub-packets for $\alpha = 0.1$ and letting $D_{sp} = [50, 100, 500, 1000, 2000]$ as shown in Figure 69. As is shown on the left plot, $D_{sp} = 500$ bytes results in the lowest latency for P1D. The additional amount of overhead required for these sub-packets does not affect the performance of the network in low-noise situations as is seen on the right plot.

Latencies that can be expected for this application with settings as described above are dependent on the size of the data packet. As can be seen in Figure 70, an increase in packet size of two orders of magnitude results in unacceptably long latencies.

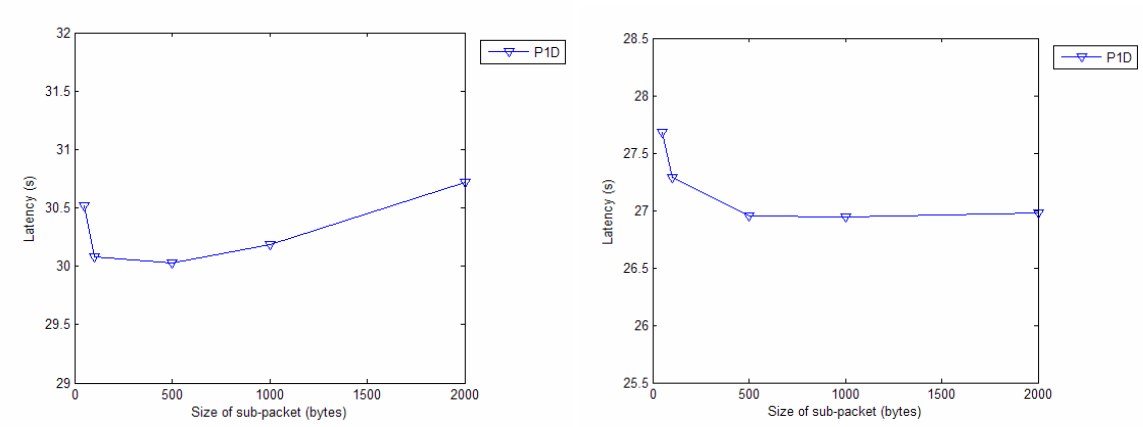

Figure 69 The optimum size of sub-packets for case study A with $\alpha = 0.1$ (left) is $D_{sp} \approx 500$ bytes and does not affect the performance in low-noise conditions ($\alpha = 0.01$, right).

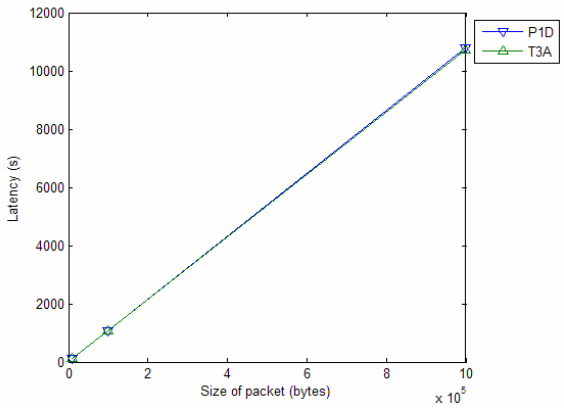

Figure 70 Latency increases linearly for $D_p = [10\text{k}, 100\text{k}, 1\text{M}]$ bytes.

We summarize the results of this case study as follows. For a Seastar application that requires high reliability and has low latency, P1D appears to be the most suitable network strategy. The simulation assumed that all nodes transmit data within a cycle and showed that a network consisting of 4 peripheral nodes is capable of realizing 2 kbytes data transmissions at realistic data rates ($R_{b1} = 3000$ bits/s and $R_{b2} = 2000$ bits/s) with a latency that remains below one minute. Transmitting 10-kbytes data packets cannot be achieved using MFSK within our latency restrictions. Packet sizes that are orders of magnitude larger in size result in unacceptably high latencies.

B. HIGH-SPEED TARGET TRACKING

To allow analysis of operational use of P1E and T2B, we formulate a variation on the first case study. Consider a dense sensor network that can be deployed rapidly with the purpose of detecting high-speed threats such as torpedoes or surface vessels. Such threats require quick reaction and do not tolerate long latencies or large data packets containing detailed target description. A node will therefore only report the detection in terms of an acoustic or electromagnetic "hit" due to the passage of a possible target. Based on the node's location, the threat direction can be discerned. The large node density allows sensor overlap (redundancy) and short latencies are preferred over occasional transmission failures. The allowed latency is set to 20 s. The default network settings can be found in Table 8.

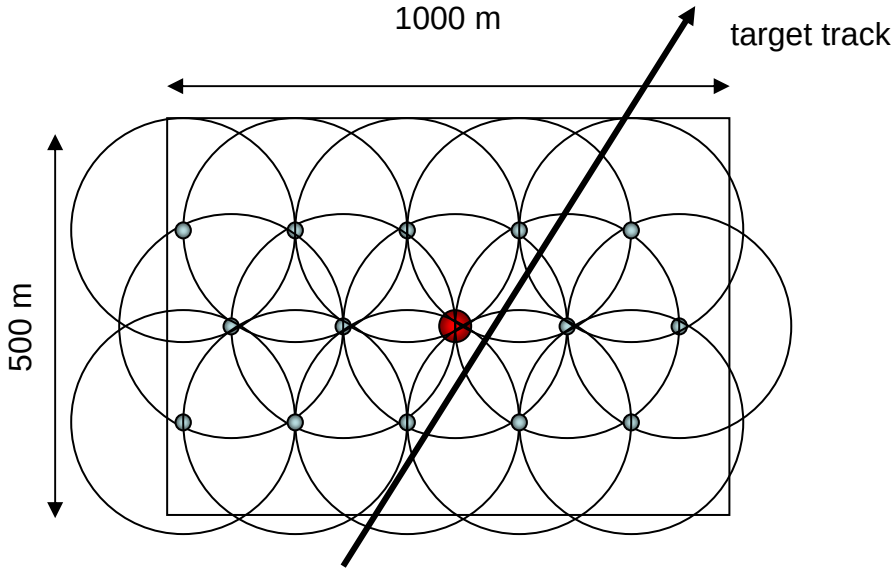

Figure 71 Schematic representation of a high density Seastar network as described in case study B.

The first parameter that we analyze, is the number of nodes with the following settings: $R_{b1} = 8000$ bits/s, $R_{b2} = 2000$ bits/s, $\alpha = 0.01$ and $n = [6, 10, 14, 18, 22]$ nodes. The channel utilization and information throughput are definitely not optimal but the values for latency look promising (see Figure 72). The token ring strategies (T2B and T3A) perform much better than the polled strategies (P1D and P1D) and latencies below 20 seconds for 14 nodes are achieved.

PARAMETER	REF	UNITS	DEFAULT
Number of modems	n	[ ]	var
wake up time	t_{wu}	[s]	0.1
acquistition time	t_{acq}	[s]	0.1
size of utility packet	d_{ut}	[bytes]	8
size of crc	d_{crc}	[bytes]	2
size of header	d_{nw}	[bytes]	12
delay poll-data	t_{d1}	[s]	0.4
delay data-poll	t_{d2}	[s]	0.4
delay manual	t_{d3}	[s]	0
delay data-SRQ	t_{d4}	[s]	0.4
time out period	t_{d5}	[s]	3

PARAMETER	REF	UNITS	DEFAULT
packet size	D_p	[bytes]	128
sub-packet size	D_{sp}	[bytes]	128
bit rate (data)	R_{b1}	[bits/s]	Var
bit rate (utility)	R_{b2}	[bits/s]	Var
maximum SRQ retries	m_{srq}	[ ]	1
maximum ACK retries	m_{ack}	[ ]	1
maximum poll retries	m_{poll}	[ ]	1
maximum token retries	m_{token}	[ ]	1
trigger level 0=min 1=max	A	[ ]	Var
simulation period	T	[hrs]	10
simulation repeats	A	[ ]	10

Table 8 Default network settings for a high-density, low-latency network (case study B).

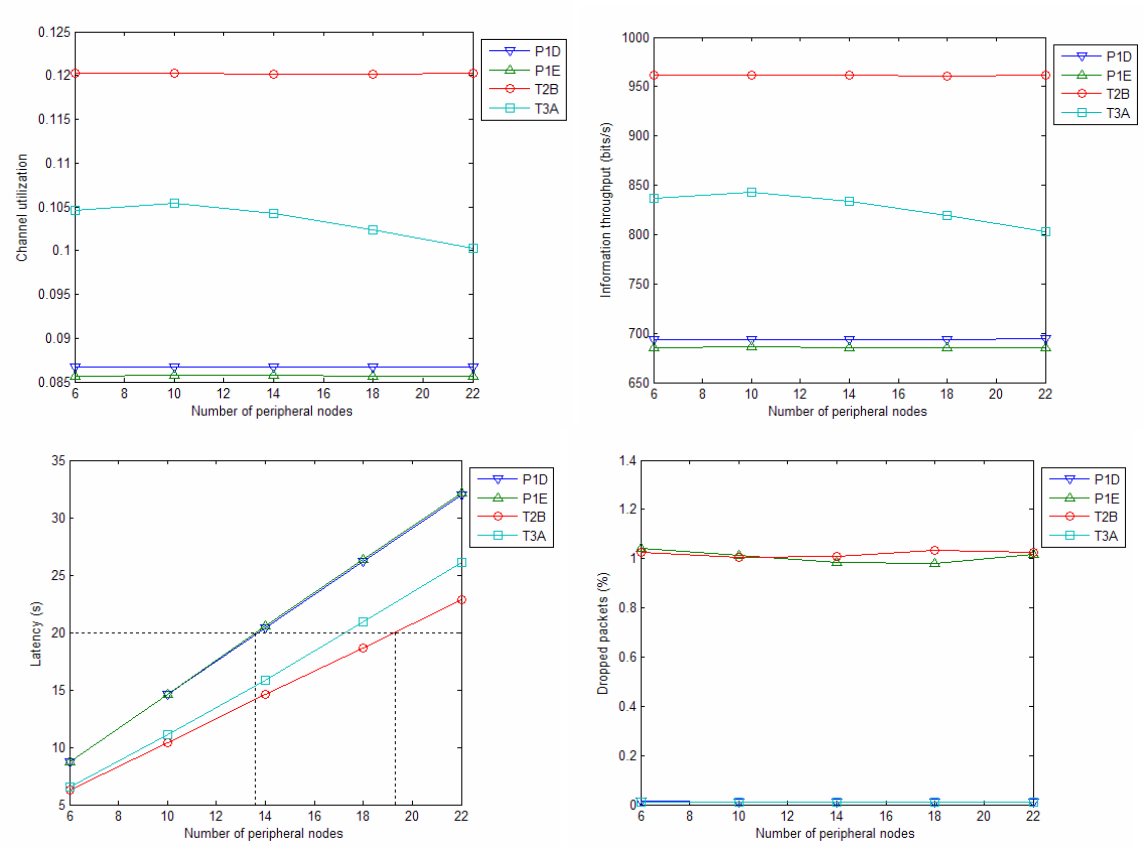

Figure 72 Effect of the number of nodes for case study B on latency for $R_{b1} = 8000$ bits/s, $R_{b2} = 2000$ bits/s and $\alpha = 0.01$ shows promising values.

Assuming that response times of 20 s are reasonable, our focus now shifts toward the required bit rate. Figure 73 shows that bit rates (R_{b1}) exceeding 2400 bits/s (T2B) and 3300 bits/s (T3A) for 14 nodes ($n = 14$) and $\alpha = 0.01$ produce latencies below 20 s, meaning that MFSK is a suitable modulation scheme under low-noise conditions. Noisy conditions, represented by $\alpha = 0.2$, result in too much distortion and latencies exceeding 25 s should be anticipated. Deploying multiple lower density clusters is a means to reduce the latency even more. Again, T2B and T3A are superior in terms of latency. Figure 74 shows the impact of the packet size on the network performance.

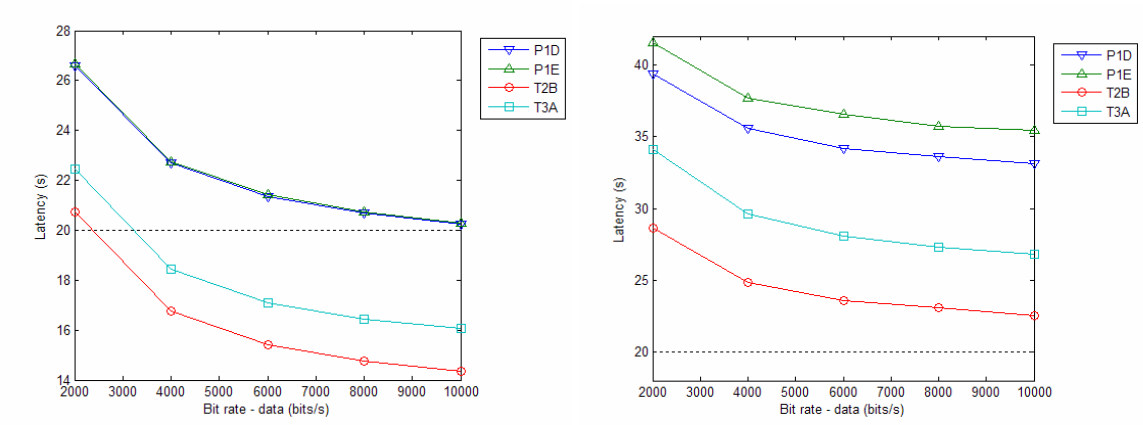

Figure 73 Latency versus bit rate (R_{b1}) for 14 nodes ($n = 14$) shown for $\alpha = 0.01$ (left) and $\alpha = 0.2$ (right). The dotted horizontal line marks the desired 20 s latency for case study B.

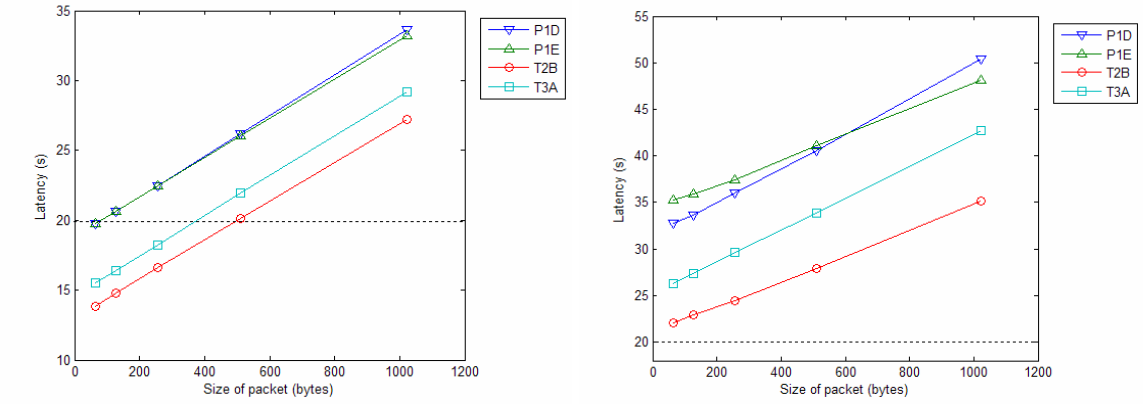

Figure 74 Latency versus packet size (D_p) for 14 nodes ($n = 14$) shown for $\alpha = 0.01$ (left) and $\alpha = 0.2$ (right). The dotted horizontal line marks the desired 20 s latency

In summary, a Seastar network that demands a very low latency at a high sensor density can only transmit very compressed data packets. T2B is by far the most suitable network strategy but suffers a large percentage of dropped packets in noisy environments. T3A is the best alternative, allowing some form of retransmission but this comes at the cost of a 15% longer latency.

C. MOBILE SWARM

The next case study involves underwater surveillance using a swarm of mobile nodes. As an example, Figure 75 depicts the use of gliders or crawlers. To allow comparison of the candidate network strategies, the condition that is imposed on this set of mobile nodes is that the swarm maintains its formation while proceeding through the water. The maximum range from central node to the most distant peripheral node is 500 m, allowing all neighboring peripheral nodes to communicate with each other as well as with the central node. The central node collects large data packets from the peripheral nodes and fuses them into more compact data sets. These compiled data sets can then either be uploaded to a fixed Seaweb node, or moored gateway buoy, when in the vicinity or transmitted via Iridium when surfaced.

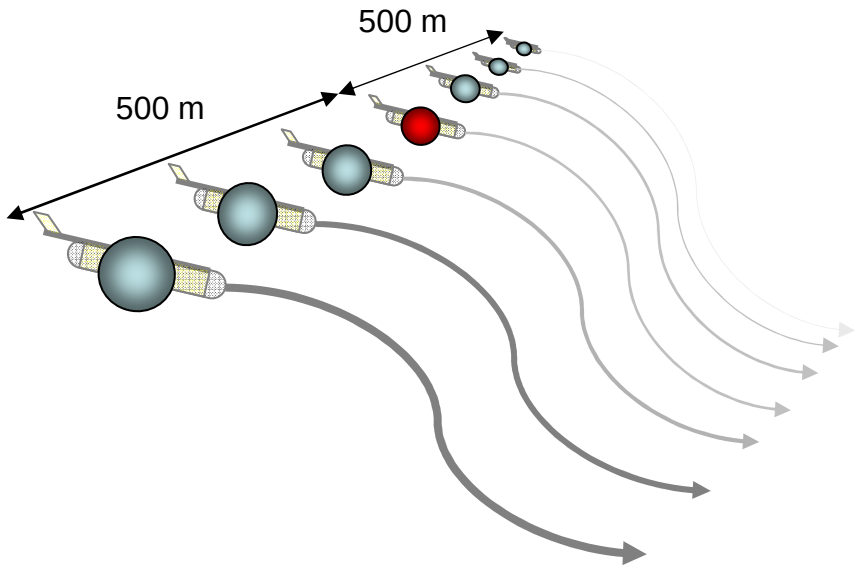

Figure 75 Swarms of UUVs, or crawlers, collecting data, forming a mobile Seastar network.

Unlike the first case study, this Seastar network is not limited by a latency requirement since data uploads from the central node are infrequent, non-real-time events. The type of data is left unspecified, but we assume large data packets ($D_p = 1$ Mbyte) that are collected over a long period and, because of the relatively large node density, an occasional lost packet is acceptable. The general settings are summarized in Table 9, where "var" refers to a set of values that are specified through simulation. Strategies P1E and T2B are not considered because of unreliable performance in noisy conditions.

PARAMETER	REF	UNITS	DEFAULT	PARAMETER	REF	UNITS	DEFAULT
Number of modems	n	[ ]	var	packet size	D_p	[bytes]	1.000.000
wake up time	t_{wu}	[s]	0.1	sub-packet size	D_{sp}	[bytes]	var
acquistition time	t_{acq}	[s]	0.1	bit rate (data)	R_{b1}	[bits/s]	var
size of utility packet	d_{ut}	[bytes]	8	bit rate (utility)	R_{b2}	[bits/s]	var
size of crc	d_{crc}	[bytes]	2	maximum SRQ retries	m_{srq}	[ ]	2
size of header	d_{nw}	[bytes]	12	maximum ACK retries	m_{ack}	[ ]	2
delay poll-data	t_{d1}	[s]	0.4	maximum poll retries	m_{poll}	[ ]	2
delay data-poll	t_{d2}	[s]	0.4	maximum token retries	m_{token}	[ ]	2
delay manual	t_{d3}	[s]	0	trigger level 0=min 1=max	A	[ ]	var
delay data-SRQ	t_{d4}	[s]	0.4	simulation period	T	[hrs]	10
time out period	t_{d5}	[s]	3	simulation repeats	A	[ ]	100

Table 9 Settings for mobile swarm case study. The value "var" refers to a variable that is an outcome of the simulation.

In order to get a feeling for information throughput and latencies with this data packet size, we first do a parametric analysis of bit rate $R_{b1} = [2000, 4000, 6000, 8000, 10000]$ bits/s, where $R_{b2} = 2000$ bits/s, $n = 6$, $D_{sp} = 10$ kbytes, and $\alpha = 0.01$. The results (see Figure 76) show latencies in the order of hours, which is unacceptably high. Even if these long latencies were allowed, they would affect the sensor sampling rate. To illustrate this, consider the following example. Sampling 1 minute of data results in a 1-Mbyte data packet. It takes two hours before all nodes have transmitted their data packets, which implies that no other data can be collected in that time frame. This reveals a complicated relation between sampling rate, packet size and bit rate which should be carefully considered.

Further increasing the bit rate by an order of magnitude is unrealistic and, instead, we reduce the packet size to $D_p = 100$ kbytes. Packet size, especially in combination with a large number of nodes, is therefore a limitation in general for a Seastar LAN. With the reduced packet size, acceptable latencies (8 to 13 minutes) are achieved for 6 to 10 kbits/s, respectively (see Figure 77) and both P1D and T3A perform well. MFSK is not sufficient to achieve these data rates and therefore other modulation schemes should be explored.

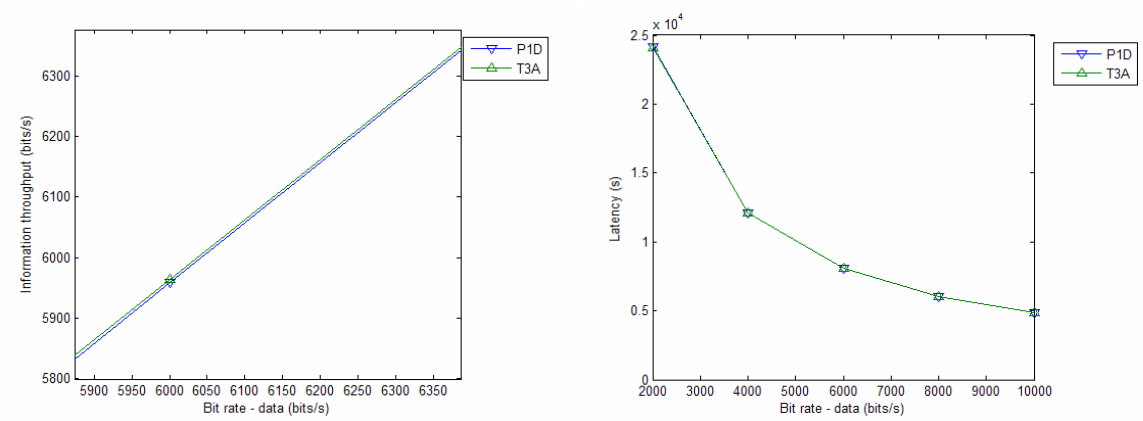

Figure 76 The effect of large data packets ($D_p = 1$ Mbyte) for case study C. The results show unacceptably high latencies for $R_{b1} = [2000, 4000, 6000, 8000, 10000]$ bits/s.

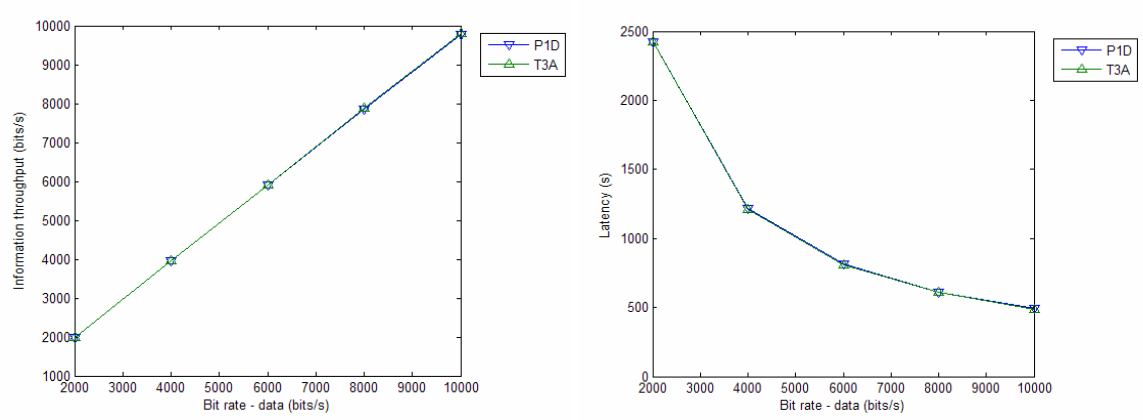

Figure 77 Reduced packet size ($D_p = 100$ kbytes) shows latencies of 8 to 13 minutes for a total of $n = 6$ peripheral nodes and $R_{b1} \geq 6000$ bits/s (case study C).

We now study the effect of adjusting the number of nodes for two cases. The first case assumes recoverable mobile peripheral nodes, allowing the implementation of more sophisticated processors that remove the asymmetry in bit rates. For this case, $R_{b1} = R_{b2} = 8000$ bits/s. The second case considers disposable mobile peripheral nodes that use $R_{b1} = 8000$ bits/s but $R_{b2} = 2000$ bits/s. Both cases assume $D_p = 100$ kbytes, $D_{sp} = 10$ kbytes, $\alpha = 0.01$ and $n = [2, 4, 6, 8, 10, 12, 14]$ nodes.

Figure 78 shows that increasing R_{b2} at these data rates does not have significant effect on the latency which is consistent with keeping the design and cost of the communications part of the peripheral nodes low. The linear relation between number of nodes and latency and causes latencies exceeding 20 minutes for $n > 14$ nodes. One possible way to overcome this is to deploy two smaller mobile Seastar networks collecting data simultaneously while keeping latencies low. This, however, would require measures to avoid interference.

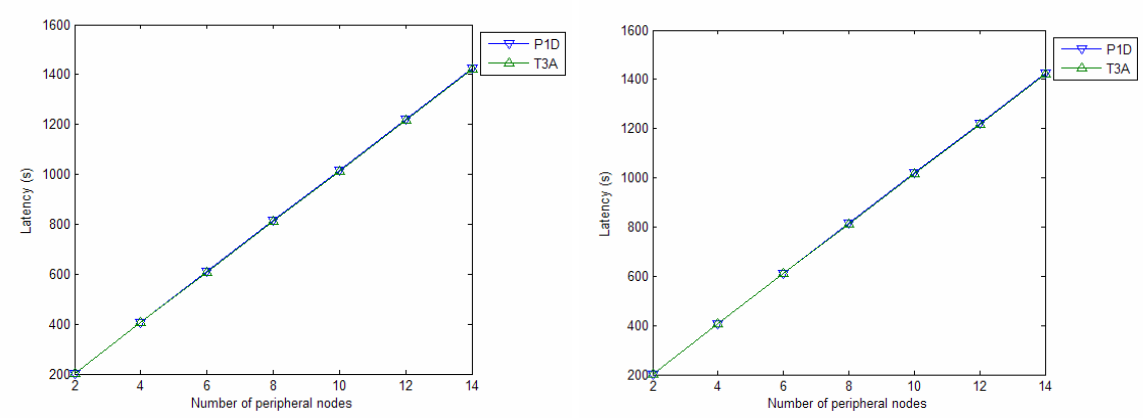

Figure 78 Latency versus number of modems for $R_{b1} = R_{b2} = 8000$ bits/s (left) and $R_{b1} = 8000$ bits/s with $R_{b2} = 2000$ bits/s (right).

The analysis is finalized by a study of performance under the influence of noise by letting $\alpha = [0, 0.01, 0.05, 0.1, 0.15, 0.2]$ for $R_{b1} = 8000$ bits/s with $R_{b2} = 2000$ bits/s and $n = 6$ nodes. Figure 79 shows that P1D performs better, as expected. The gain of accepting 4% more dropped packets at $\alpha = 0.2$ when choosing for T3A instead of P1D is

a 10% reduction in latency, recognizing that the simulation generates a delaying artificial packet-out-of-sequence situation that may not be realistic.

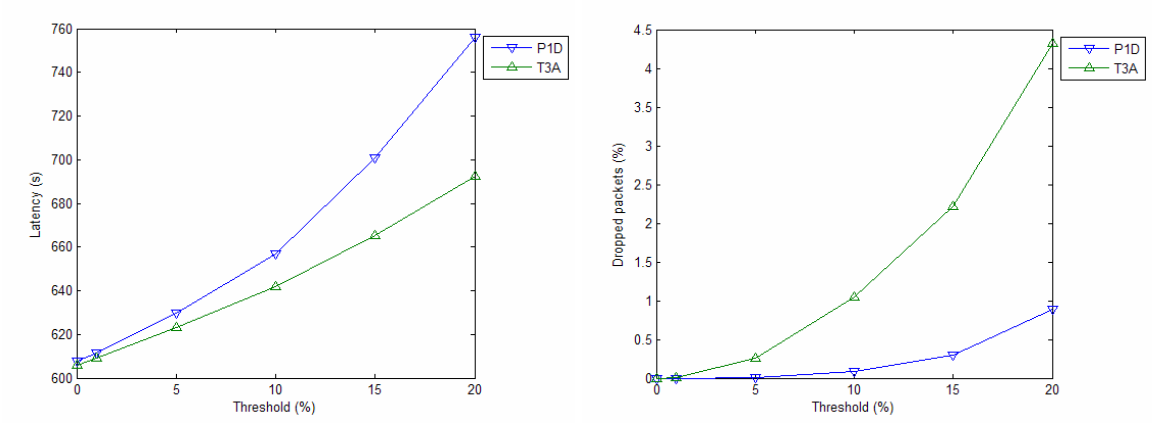

Figure 79 Latency (left) and dropped packets (right) versus increasing noise when P1D is allowed only 2 retransmissions.

Optimizing for the size of sub-packets gives only a 1% improvement in network performance (see Figure 80). In order to avoid abundant overhead in low-noise conditions D_{sp} should not be less than 2 kbytes.

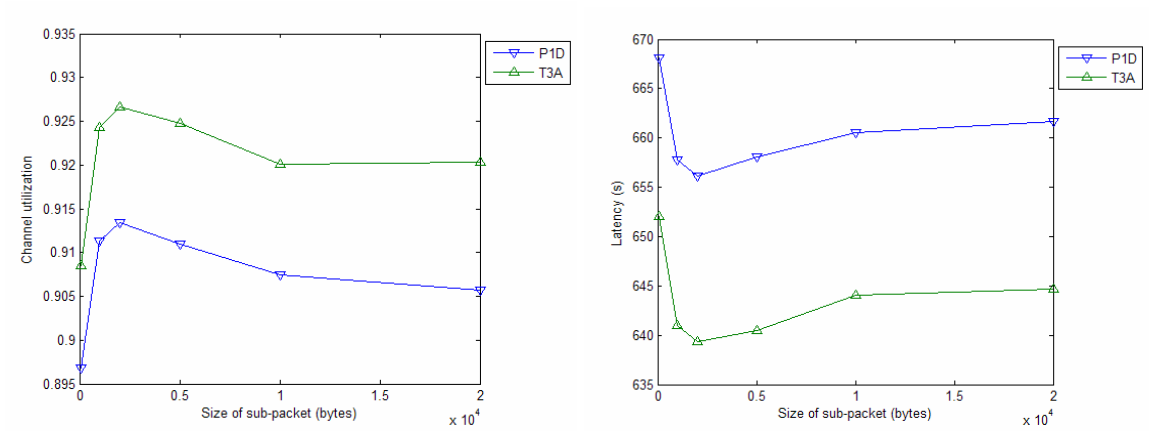

Figure 80 The effect of varying the size of sub-packets on channel utilization (left) and latency (right) for $\alpha = 0.1$.

This case study reveals that the latency requirement imposes a restriction on the allowed size of data packets. It also reveals a complex relation between sampling rate, bit

rate, packet size and number of nodes. These issues can not be solved by further increasing bit rates because the available bandwidth is limited and more sophisticated modulation schemes, such as PSK or OFDM, are not yet found reliable enough for practical purposes. Solutions have to be found at the application layer in terms of data compression, or through the use of multiple Seastar clusters.

For this case study, P1D proved to be the most reliable network strategy. The additional gain in terms of latency that T3A gives only shows up strongly under noisy conditions. Taking into account the additional geometry restriction that T3A brings along, the optimum strategy in a mobile Seastar network would be P1D.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS

A. RESULTS

For a given transmission range of 500 m, a link budget analysis has shown that, in the presence of wind, an optimum carrier frequency can be found near 41 kHz. A minimum *SL* of 106 dB re 1 μPa @ 1 m is required to achieve a $SNR_a = 1$ at the receiver.

MFSK is the most robust, readily available modulation technique for Seastar underwater acoustic links. An analytical expression was developed to evaluate the relation between bandwidth, optimum number of bits/symbol and bit rate. The outcome of this evaluation depends on the bandwidth definition of a sinc-function. From a hardware perspective, the useable bandwidth BW_x and bit rate R_b are limited by the physical properties of the transducer but a BW_x of 20 kHz, centered near a carrier frequency of 41 kHz, resulting in maximum R_b of 3300 bits/s should be achievable. It was theoretically shown that OFDM could increase R_b by an order of magnitude for the same BW_x. The energy budget for central and peripheral modems is asymmetric in the sense that the energy of the central node required to operate the network is two orders of magnitude larger than that of a peripheral node.

A prototype Seastar was implemented using available Seaweb modems. Experiments were performed, both in air and water, with the Seastar prototype using a polling strategy. The experiments show that the concept is viable and that the prototype is able to operate persistently without human intervention. Integration of the polling algorithm, reduction of delays and a smart design of the polling utility packet would increase the performance significantly. New modem hardware capable of transmitting at higher bit rates would further increase the performance.

The network simulations gave an insight into the effect that certain network parameters have on the performance. For applications that require low latency and accept low transmission reliability, a token strategy without retransmissions is the best candidate. In general, however, reliable communications are required and therefore SRQ-

capable network strategies are preferred. Polling with transmission shows the best performance in terms of reliability whereas token-based networking with retransmission performs slightly better in terms of latency and information throughput and channel utilization.

Case studies confirm the above conclusions and show at the same time that *the* optimum Seastar network strategy does not exist. Network optimization fully depends on the operational requirements and boundary conditions that are imposed upon the network.

B. RECOMMENDATIONS FOR FUTURE WORK

Future research should focus on questions that have not been addressed in this thesis such as the integration of the Seastar LANs with Seaweb and prevention of inter-LAN interference. A study that includes geometry and the use of mobile nodes may generate new ideas or network strategies.

In-depth studies in future modulation types such as OFDM, PSK and QAM that may apply to underwater acoustic links are available but do not yet show the desired results for Seastar application in water. If these techniques would show MFSK-like robustness, the network performance in terms of bandwidth efficiency and achievable bit rates would greatly improve. Developments in this field should be closely followed.

Further in-water and in-air experiments under varying conditions and in different test environments could improve the simulation model that was used and provide more accurate data regarding network performance. A calibration between measured noise levels and simulated trigger levels would allow more realistic performance.

C. IMPACT

Although this thesis revealed some Seastar limitations in terms of packet size, bit rate and latency, the Seastar concept has the potential to increase node density at affordable cost to the US Navy Seaweb wide area network.

APPENDIX A. POLLING ALGORITHM DEVELOPED IN C

This polling algorithm is a modification to a code for allowing communications with Seaweb modems that was originally developed by Chris Fletcher from SPAWAR Systems Center, San Diego. In order to reduce the amount of code in this thesis, only the modified part containing the essentials of the polling algorithm is shown.

```
/* Polling Algorithm for Seastar Prototype
/* Editor: Bjorn Kerstens
/* Apr-Jun 2007

/* Determine input availibity*/
   int result,keyboard_read;

/* Determine polling variables
   Assume central hub = address 1, assume peripheral hubs have addresses 2-nrperiph*/
   int i,j,cmdsize,length;

   time_t start,stop, startcrashdelay, stopcrashdelay;
   double delay, crashdelay;

   /*double delay;*/
   char cmd[8];
   char *checkstr1, *checkstr2, *checkstr3, *errorstr1, *errorstr2, ...
   *errorstr3, *errorstr4, *crashstr1, *crashstr2, *crashstr3, ...
   *lowpstr1, *lowpstr2, *lowpstr3 ;

/* Initialize Inputs*/

   fd_set inputs;
   fd_set stdin_input;
   struct timeval timeout;

FD_ZERO(&inputs);

while (1)
   {     /* loop forever until CTRL-C*/

/*This part starts the polling sequence*/
for(i=3;i<8;i++)
     {
```

```
        /* Set timer*/
         timeout.tv_sec = 0;
         timeout.tv_usec = 500;

         FD_SET(0,&inputs);

         result = select(1,&inputs,NULL,NULL,&timeout);

         /* Check for Errors */
      if (result < 0)
        {
         perror("select failed");
         exit(-1);
        }
       }
/* This part contains the polling algorithm*/
      else if (result == 1)
        {
              delay=0;
        cmdsize=sprintf(cmd,"at$bt%d\r\n",i);

         if ((atoi(argv[2]) == 1))
           {
            write(STDOUT,&cmd,cmdsize);
           }

           write(fd,&cmd,cmdsize);
           write(fs,&cmd,cmdsize);

           checkstr1 = NULL;
           checkstr2 = NULL;
           checkstr3 = NULL;

           errorstr1 = NULL;
           errorstr2 = NULL;
           errorstr3 = NULL;
           errorstr4 = NULL;

           crashstr1 = NULL;
           crashstr2 = NULL;
           crashstr3 = NULL;

          lowpstr1 = NULL;
          lowpstr2 = NULL;
```

```
        lowpstr3 = NULL;

/* This part looks for strings CRC:Pass, Aborted or abort to determine success or failure of a transmission*/

        while(checkstr1 == NULL && checkstr2 == NULL && checkstr3 == NULL ...
    && errorstr1 == NULL && errorstr2 == NULL && errorstr3 == NULL ...
    && errorstr4 == NULL && crashstr1 == NULL && crashstr2 == NULL ...
    && crashstr3 == NULL && lowpstr1 == NULL && lowpstr2 == NULL && lowpstr3 == NULL)
        {
            start=time(0);
            res=read(fd,buf,255);
            buf[res]='\0';

       if ((atoi(argv[2]) == 1))
       {
        write(STDOUT,&buf,res);
                }

                        write(fs,&buf,res);

                        checkstr1 = (char *)strstr(buf,"C:P");
                        checkstr2 = (char *)strstr(buf,":Pa");
                        checkstr3 = (char *)strstr(buf,"Pas");

                        errorstr1 = (char *)strstr(buf,"Abo");
                        errorstr2 = (char *)strstr(buf,"bor");
                        errorstr3 = (char *)strstr(buf,"ort");
                        errorstr4 = (char *)strstr(buf,"abo");

                        crashstr1 = (char *)strstr(buf,"WAI");
                        crashstr2 = (char *)strstr(buf,"AIT");
                        crashstr3 = (char *)strstr(buf,"IT_");

                        lowpstr1 = (char *)strstr(buf,"owp");
                        lowpstr2 = (char *)strstr(buf,"wpo");
                        lowpstr3 = (char *)strstr(buf,"pow");

    if (crashstr1 || crashstr2 || crashstr3 || lowpstr1 || lowpstr2 || lowpstr3)
                {
                crashdelay=0;
                startcrashdelay=time(0);
                while (crashdelay<600)
                    {
```

```
                    res=read(fd,buf,255);
                    buf[res]='\0';

                    if ((atoi(argv[2]) == 1))
                      {
                       write(STDOUT,&buf,res);
                      }
                    write(fs,&buf,res);
                    stopcrashdelay=time(0);
                    crashdelay=difftime(stopcrashdelay,startcrashdelay);
                   }
                }
  }
/* Time delay to prevent cross talk after " *!*Packet OOS" situation*/
                while(delay<10)
                {
                        res=read(fd,buf,255);
                        buf[res]='\0';
                        string = (char *)strstr(buf,">");
/*This part looks in buf for the modem delimiter ">" and sends a time stamp to STDOUT
and the datafile*/
                        if (string)
                        {
                                (void)time(&timeval);

                                 if ((atoi(argv[2]) == 1))
                                {
                                        write(STDOUT,ctime(&timeval),26);
                                 }

                                write(fs,ctime(&timeval),26);
                        }

                        if ((atoi(argv[2]) == 1))
                        {
                         write(STDOUT,&buf,res);
                        }

                        write(fs,&buf,res);
                        stop=time(0);
                        delay=difftime(stop,start);
                }
        }

  }
```

APPENDIX B. NETWORK SIMULATION ALGORITHMS

The function INI.m is used to insert data for simulation and set parameters. The bracketed values are set up to contain arrays.

```
function [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI

   number_modems=[14];  % number of peripheral modems       def=6

   t_wu=0.1;      % duration of wake-up tones [s].          default=0.4
   t_acq=0.1;     % duration of acquisition tones [s].      default=0.28
   d_ut=8;        % bytes in utility packet.                default=9
   d_crc=2;       % bytes in CRC.                           default=2
   d_nw=12;       % bytes in network header.                default=14

   t_delay1=0.4;  % [s] delay measured btwn poll-data    default =1
   t_delay2=0.4;  % [s] delay measured after data xmit   default =2.9
   t_delay3=0;    % [s] additional delay in prototype    default =0
   t_delay4=0.4;  % [s] delay between data-SRQ order     default =0.7
   t_delay5=3;    % [s] (S7) - acoustic response timeout default =7.5

   data_set=[64 128 256 512 1024];    % length of single datapacket (information) to be
transmitted [bytes]
   L_sp=[128];                          % length of subpacket [bytes]    default=256

   baudrate1=[8000];    % [bps]                          default =800
   baudrate2=[2000];    % [bps]                          default =140
   maxretries=[1];      % maximum number of retries      default =3
   maxCTS=1;            % (not used)                     default =1
   maxACK=[1];          % maximum number of ACK retries      default =3
   maxPOLL=[1];         % maximum number of POLL retries     default =3
   maxTOKEN=[1];        % maximum number of TOKEN retries    default =3

   A=100   % amount of times that the simulation will be executed...
           %  in order to get reliable averages          default =100
   T=10;       % simulation period [hrs]                 default =10
           %  (e.g T=1 means simulate network for a period of 1 hour)

   alfa=[0.2];   % trigger level   (0-1)              def=0.05

   if alfa == 0
      L1 = 0, L2 = 0, L3 = 0, L23(m3) = 0, L4(m3) = 0;
   else
      L1=1;      L2=1-alfa.*1;      L3=1-alfa.*2;      L23=alfa;
      L4=alfa;   L5=0;
   end
```

The function COLLECTDATA.m executes the simulation and collects the outcome under the variable name “results”.

```
function[results]=COLLECTDATA

results=[POLL1d POLL1e TOKEN2b TOKEN3a];
```

The function PLOTDATA.m takes the output of COLLECTDATA.m and creates figures to display the results. It summarizes the input parameters that were used for the plots under the variable “settings.” Figures can be plotted for one parameter or two parameters at the same time.

```
function[settings]=PLOTDATA(results)

[number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;
settings=[number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN];
clf;

% Run COLLECTDATA function FIRST !!!

DPARA=0;
xname = 0;
yname = 0;
tag={' POLL1d' ' POLL1e' ' TOKEN2b' ' TOKEN3a'};

% split up results matrix

SIZE = size(results);
SIZEX = SIZE(2)/7;
SIZEY = SIZE(1)/4;

% determine single parameter or double parameter plot

if SIZEX > 1
   DPARA = 1;
end

if DPARA

   if length(baudrate1)>1
      xname= 'Bit Rate (bits/s)';
```

```
    xdata = baudrate1;
end

if length(number_modems)>1
    if xname==0
        xname= 'Number of Modems';
        xdata = number_modems;
    else
        yname= 'Number of Modems';
        ydata = number_modems;
    end
end

if length(data_set)>1
    if xname==0
        xname= 'Size of Packet (bytes)';
        xdata = data_set;
    else
        yname= 'Size of Packet (bytes)';
        ydata = data_set;
    end
end

if length(L_sp)>1
   if xname==0
        xname= 'Size of Subpacket (bytes)';
        xdata = L_sp;
    else
        yname= 'Size of Subpacket (bytes)';
        ydata = L_sp;
   end
end

if length(maxretries)>1
   if xname==0
        xname= 'Number of POLL/TOKEN/SRQ/ACK retries';
        xdata = maxretries;
    else
        yname= 'Number of POLL/TOKEN/SRQ/ACK retries';
        ydata = maxretries;
   end
end

if length(baudrate2)>1
   if xname==0
        xname= 'Low Bit Rate (bits/s)';
        xdata = baudrate2;
    else
        yname= 'Low Bit Rate (bits/s)';
        ydata = baudrate2;
   end
end
```

```
if length(t_delay3)>1
   if xname==0
       xname= 'Built-in Delay (s)';
       xdata = t_delay3;
    else
       yname= 'Built-in Delay (s)';
       ydata = t_delay3;
    end
end

xdata
ydata

for i=1:3
ii=i;
if ii>5;
    ii=ii+1;
end

figure(1)
clf;
    subplot(3,3,ii)
    pcolor(ydata,xdata,results(1:SIZEY,(i-1)*SIZEX+1:i*SIZEX));
    shading interp
    xlabel(yname);
    ylabel(xname);
    caxis([0.2 0.4]);
    colorbar;
    title(strcat('Channel Utilization' , tag(i)));

figure(2)
clf;
    subplot(3,3,ii)
    pcolor(ydata,xdata,results(SIZEY+1:2*SIZEY,(i-1)*SIZEX+1:i*SIZEX));
    %shading interp
    xlabel(yname);
    ylabel(xname);
    %caxis([0.2 0.4]);
    colorbar;
    title(strcat('Information Throughput' , tag(i)));

figure(3)
clf;
    subplot(3,3,ii)
    pcolor(ydata,xdata,results(2*SIZEY+1:3*SIZEY,(i-1)*SIZEX+1:i*SIZEX));
    shading interp
    xlabel(yname);
    ylabel(xname);
    %caxis([0.2 0.4]);
    colorbar;
    title(strcat('Latency' , tag(i)));

figure(4)
```

```
    clf;
        subplot(3,3,ii)
        pcolor(ydata,xdata,results(3*SIZEY+1:4*SIZEY,(i-1)*SIZEX+1:i*SIZEX));
        %shading interp
        xlabel(yname);
        ylabel(xname);
        caxis([0 50]);
        colorbar;
        title(strcat('Dropped Packets' , tag(i)));
  end

else

    if length(baudrate1)>1
        xname = 'Bit rate - data (bits/s)';
        xdata = baudrate1;
    elseif length(number_modems)>1
       xname = 'Number of modems';
       xdata = number_modems;
    elseif length(data_set)>1
       xname= 'Size of packet (bytes)';
       xdata = data_set;
    elseif length(L_sp)>1
       xname= 'Size of sub-packet (bytes)';
       xdata = L_sp;
    elseif length(maxretries)>1
       xname= 'Number of POLL/TOKEN/SRQ/ACK retries';
       xdata = maxretries;
    elseif length(baudrate2)>1
       xname= 'Bit rate - utility packets (bits/s)';
       xdata = baudrate2;
    elseif length(L2)>1
       xname= 'Trigger level (%)';
       xdata = alfa*100;
    end

    figure(1)
    clf;
    plot(xdata,results(1:SIZEY,:),'-o');
    legend('P1D','P1E','T2B','T3A',-1);
    xlabel(xname);
    ylabel('Channel utilization');

    figure(2)
    clf;
    plot(xdata,results(SIZEY+1:2*SIZEY,:),'-o');
    legend('P1D','P1E','T2B','T3A',-1);
    xlabel(xname);
    ylabel('Information throughput (bits/s)');

    figure(3)
    clf;
    plot(xdata,results(2*SIZEY+1:3*SIZEY,:),'-o');
    legend('P1D','P1E','T2B','T3A',-1);
```

```
    xlabel(xname);
    ylabel('Latency (s)');

    figure(4)
    clf;
    plot(xdata,results(3*SIZEY+1:4*SIZEY,:),'-o');
    legend('P1D','P1E','T2B','T3A',-1);
    xlabel(xname);
    ylabel('Dropped packets (%)');

end
```

The function POLL1d.m is used to simulate network strategy P1D.

```
function[result1d, energy1d]=POLL1d

clear,

%************************************************************************
%
% Initiate values through <ini.m> function
%
%************************************************************************
[number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
    t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
    L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

for m1=1:length(baudrate1)      % baudrate1
for m2=1:length(data_set)       % data length
for m3=1:length(L2)             % noise
for m4=1:length(maxretries)     % maxretries
for m5=1:length(L_sp)           % subpacket length
for m6=1:length(number_modems)  % number_modems
for m7=1:length(baudrate2)      % baudrate2

    % calculate # subpackets and duration of utility packet

    N_sp(m5)=ceil(data_set(m2)/L_sp(m5));       %number of subpackets
    t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;    % transmission time of utility packet

%************************************************************************
%
% Do the simulation a couple of times to get a good average of the
% metrics you're interested in
%
%************************************************************************

  for a=1:A;        % run simulation 'a' times

% reset all values to zero
```

```
OOS=0;
DOOS=0;
OOSFLAG(1:number_modems(m6))=0;
OOSFLAG1=0;
OOSFLAG2=0;
REPOLL=1;

d_total=0;
t_total=0;
latency1=0;
latency(a)=0;
pktabort(a)=0;
pktcorrect(a)=0;

d_modem1(1:number_modems(m6),1)=0;
d_modem2(1:number_modems(m6),1)=0;

t_modem1(1:number_modems(m6),1)=0;
t_modem2(1:number_modems(m6),1)=0;

cycle=1;

%**********************
%energy calc. only !!!
TP=0;
TI=0;
Wrcv=0.5;   %[W]
Wxmt=0.1;   %[W]
%**********************

while t_total<(3600*T)

   for address=1:number_modems(m6);

%*********************************************************************
% In case OOSFLAG flag is set to 1, the transmission consists of the new
% packet sent at bit rate1 and the old packet at bit rate2
%*********************************************************************

      if OOSFLAG(address)

         [OOS,DOOS,OOSFLAG1]=oos(m1,m2,m3,m4,m5,m6,m7,...
            OOSFLAG(address),REPOLL);

         OOSFLAG(address) = OOSFLAG1;

      end

      %***********************************************************
      % info packet
      %***********************************************************
```

```
[INFO,DATA]=info(m1,m2,m3,m4,m5,m6,m7,REPOLL);

dice1=rand(1);

%*********************************************************
% utility or info packet corrupted
%*********************************************************

[SRQ,DATA,OOSFLAG2]= srq1(m1,m2,m3,m4,m5,m6,m7,dice1,...
   OOSFLAG2,REPOLL);

OOSFLAG(address) = OOSFLAG2;

%*********************************************************
% POLLING utility packet
%*********************************************************

[POLL,POLLFLAG]=poll(m1,m2,m3,m4,m5,m6,m7,dice1,REPOLL);

      if POLLFLAG
         INFO=0; OOS=0;
         DATA=0; DOOS=0;
         OOSFLAG(address)=1;
      end

%*********************************************************
% add all events
%*********************************************************

t_xmit=POLL+INFO+SRQ+OOS;
data(m2)=DATA+DOOS;

if (DATA==0) & (POLLFLAG==0)
   pktabort(a)=pktabort(a)+1;
elseif DATA~=0;
   pktcorrect(a)=pktcorrect(a)+1;
end

t_modem1(address,cycle+1)=t_modem1(address,cycle)+t_xmit;
t_modem2(address,cycle+1)=t_xmit;
d_modem1(address,cycle+1)=d_modem1(address,cycle)+data(m2)*8;
d_modem2(address,cycle+1)=data(m2)*8;

DATA=data_set(m2);        %reset data
OOS=0;                    %reset OOSFLAG correction
DOOS=0;                   %reset OOSFLAG correction
OOSFLAG1=0;
OOSFLAG2=0;

counter=counter+1;

%*********************************************************
```

```
    % ENERGY BUDGET: ONLY IN NOISE FREE SIMULATION!!!!!!
    % Instruction: set A=1; T=1; Only for single input parameter
    %***********************************************
%       TP = TP + POLL - t_delay1;
%       TI = TI + INFO - (t_delay2+t_delay3);
%       Prcv = TP*Wrcv;
%       Pxmt = (TI)*Wxmt;
%       Phr = (Prcv+Pxmt)/number_modems(m6);
%       Crcv = (TI)*Wrcv;
%       Cxmt = TP*Wxmt;
%       Chr = (Crcv+Cxmt);
%       energy1a=[Phr; Chr];
    %*****************************************************

      end    % end "for" loop that determines modem addresses

      t_total(1,cycle+1)=t_total(1,cycle)+sum(t_modem2(:,cycle+1));
      d_total(1,cycle+1)=d_total(1,cycle)+sum(d_modem2(:,cycle+1));
      latency1(1,cycle)=sum(t_modem2(:,cycle+1));

      cycle=cycle+1;

   end       % end "while" loop - period that network is evaluated

   %*********************************************************************
   % utilization=t_information/t_total
   %
   % throughput = actual baudrate so average actual amount of bits you can
   %      transfer from peripheral to central considering present settings
   %
   % latency = the maximum period that you should have to wait to receive
   %      data from a specific address
   %*********************************************************************

   utilization(a)=(d_total(1,length(d_total))/baudrate1(m1)) / ...
      (t_total(1,length(t_total)));

   throughput(a)=(d_total(1,length(d_total))) / ...
      (t_total(1,length(t_total)));

   latency(a)=mean(latency1);

   pktabort(a)=100*pktabort(a)/( pktcorrect(a)+pktabort(a) );

  end

util_ave(m1,m2,m3,m4,m5,m6,m7)=mean(utilization);

throughput_ave(m1,m2,m3,m4,m5,m6,m7)=mean(throughput);

latency_ave(m1,m2,m3,m4,m5,m6,m7)=mean(latency);
```

```
pktabort_ave(m1,m2,m3,m4,m5,m6,m7)=mean(pktabort);

end
end
end
end
end
end
end

util_ave=squeeze(shiftdim(util_ave));
throughput_ave=squeeze(shiftdim(throughput_ave));
latency_ave=squeeze(shiftdim(latency_ave));
pktabort_ave=squeeze(shiftdim(pktabort_ave));
result1d=[util_ave; throughput_ave; latency_ave; pktabort_ave];
```

The function POLL1e.m is used to simulate network strategy P1E.

```
function[result1e, energy1e]=POLL1e

clear,

%************************************************************************
%
% Initiate values through <ini.m> function
%
%************************************************************************
[number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

for m1=1:length(baudrate1)      % baudrate1
for m2=1:length(data_set)       % data length
for m3=1:length(L2)             % noise
for m4=1:length(maxretries)     % maxretries
for m5=1:length(L_sp)           % subpacket length
for m6=1:length(number_modems)  % number_modems
for m7=1:length(baudrate2)      % baudrate2

   % calculate # subpackets and duration of utility packet

   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));    %number of subpackets
   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;    % transmission time of utility packet

%************************************************************************
%
% Do the simulation a couple of times to get a good average of the
% metrics you're interested in
%
```

```
%************************************************************************

 for a=1:A;        % run simulation 'a' times

% reset all values to zero

   REPOLL=0;

   d_total=0;
   t_total=0;
   latency1=0;
   latency(a)=0;
   pktabort(a)=0;
   pktcorrect(a)=0;

   d_modem1(1:number_modems(m6),1)=0;
   d_modem2(1:number_modems(m6),1)=0;

   t_modem1(1:number_modems(m6),1)=0;
   t_modem2(1:number_modems(m6),1)=0;

   cycle=1;

   %**********************
   %energy calc. only !!!
   TP=0;
   TI=0;
   Wrcv=0.5;   %[W]
   Wxmt=0.1;   %[W]
   %**********************

   while t_total<(3600*T)

      for address=1:number_modems(m6);

         %************************************************************
         % info packet
         %************************************************************

         [INFO,DATA]=info(m1,m2,m3,m4,m5,m6,m7,REPOLL);

         %************************************************************
         % utility or info packet corrupted
         %************************************************************

         dice1=rand(1);

         if ((dice1<L2(m3)) & (dice1>=L3(m3)))
            DATA=0;
         end
```

```
%*************************************************************
% POLLING utility packet
%*************************************************************

[POLL,POLLFLAG]=poll(m1,m2,m3,m4,m5,m6,m7,dice1,REPOLL);

    if POLLFLAG

       POLL = 2*POLL;

       dice4=rand(1);
        if dice4 < L4(m3)
           INFO=0; DATA=0;
        end
    end

%*************************************************************
% add all events
%*************************************************************

t_xmit=POLL+INFO;
data(m2)=DATA;

   if DATA==0
      pktabort(a)=pktabort(a)+1;
   else
      pktcorrect(a)=pktcorrect(a)+1;
   end

t_modem1(address,cycle+1)=t_modem1(address,cycle)+t_xmit;
t_modem2(address,cycle+1)=t_xmit;
d_modem1(address,cycle+1)=d_modem1(address,cycle)+data(m2)*8;
d_modem2(address,cycle+1)=data(m2)*8;

DATA=data_set(m2);        %reset data

counter=counter+1;

%*************************************************************
% ENERGY BUDGET: ONLY IN NOISE FREE SIMULATION!!!!!!
% Instruction: set A=1; T=1; Only for single input parameter
%*************************************************
%       TP = TP + POLL-t_delay1;
%       TI = TI + INFO - (t_delay2+t_delay3);
%       Prcv = TP*Wrcv;
%       Pxmt = (TI)*Wxmt;
%       Phr = (Prcv+Pxmt)/number_modems(m6);
%       Crcv = (TI)*Wrcv;
%       Cxmt = TP*Wxmt;
%       Chr = (Crcv+Cxmt);
%       energy1c=[Phr; Chr];
%*************************************************************
```

```
      end     % end for loop that determines modem addresses

      t_total(1,cycle+1)=t_total(1,cycle)+sum(t_modem2(:,cycle+1));
      d_total(1,cycle+1)=d_total(1,cycle)+sum(d_modem2(:,cycle+1));
      latency1(1,cycle)=sum(t_modem2(:,cycle+1));

      cycle=cycle+1;

   end        % end while loop - period that network is evaluated

   %**********************************************************************
   % utilization=t_information/t_total
   %
   % throughput = actual baudrate so average actual amount of bits you can
   %       transfer from peripheral to central considering present settings
   %
   % latency = the maximum period that you should have to wait to receive
   %       data from a specific address
   %**********************************************************************

   utilization(a)=(d_total(1,length(d_total))/baudrate1(m1)) / ...
      (t_total(1,length(t_total)));

   throughput(a)=(d_total(1,length(d_total))) / ...
      (t_total(1,length(t_total)));

   latency(a)=mean(latency1);

   pktabort(a)=100*pktabort(a)/( pktcorrect(a)+pktabort(a) );

 end

util_ave(m1,m2,m3,m4,m5,m6,m7)=mean(utilization);

throughput_ave(m1,m2,m3,m4,m5,m6,m7)=mean(throughput);

latency_ave(m1,m2,m3,m4,m5,m6,m7)=mean(latency);

pktabort_ave(m1,m2,m3,m4,m5,m6,m7)=mean(pktabort);

end
end
end
end
end
end
end

util_ave=squeeze(shiftdim(util_ave));
throughput_ave=squeeze(shiftdim(throughput_ave));
```

```
latency_ave=squeeze(shiftdim(latency_ave));
pktabort_ave=squeeze(shiftdim(pktabort_ave));
result1e=[util_ave; throughput_ave; latency_ave; pktabort_ave];
```

The function TOKEN2b.m is used to simulate network strategy T2B.

```
function[result2b, energy2b]=TOKEN2b

clear,

%************************************************************************
%
% Initiate values through <ini.m> function
%
%************************************************************************
[number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

for m1=1:length(baudrate1)      % baudrate1
for m2=1:length(data_set)       % data length
for m3=1:length(L2)             % noise
for m4=1:length(maxretries)     % maxretries
for m5=1:length(L_sp)           % subpacket length
for m6=1:length(number_modems)  % number_modems
for m7=1:length(baudrate2)      % baudrate2

   % calculate # subpackets and duration of utility packet

   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));    %number of subpackets
   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;    % transmission time of utility packet

%************************************************************************
%
% Do the simulation a couple of times to get a good average of the
% metrics you're interested in
%
%************************************************************************

 for a=1:A;        % run simulation 'a' times

% reset all values to zero

   RETOKEN=0;
   HUB=0;

   d_total=0;
   t_total=0;
   latency1=0;
```

```
latency(a)=0;
pktabort(a)=0;
pktcorrect(a)=0;

d_modem1(1:number_modems(m6),1)=0;
d_modem2(1:number_modems(m6),1)=0;

t_modem1(1:number_modems(m6),1)=0;
t_modem2(1:number_modems(m6),1)=0;

cycle=1;

%**********************
%energy calc. only !!!
TT=0;
TI=0;
Wrcv=0.5;   %[W]
Wxmt=0.1;   %[W]
%**********************

while t_total<(3600*T)

   for address=1:number_modems(m6);

      %************************************************************
      % info packet
      %************************************************************
      [INFO,DATA]=info(m1,m2,m3,m4,m5,m6,m7,RETOKEN);
      %[INFO,DATA]=info(m1,m2,m3,m4,m5,m6,m7);

      %************************************************************
      % utility or info packet corrupted
      %************************************************************

      dice1=rand(1);

      if ((dice1<L2(m3)) & (dice1>=L3(m3)))
         DATA=0;
      end

      %************************************************************
      % TOKEN utility packet
      %************************************************************

      [TOKEN,TOKENFLAG]=token(m1,m2,m3,m4,m5,m6,m7,dice1,RETOKEN,HUB);

         TOKEN = TOKEN - t_delay1;

           if TOKENFLAG
              INFO=0;
              DATA=0;
           end
```

```
        %*************************************************************
        % add all events
        %*************************************************************

        t_xmit=TOKEN+INFO;
        data(m2)=DATA;

        if DATA==0
           pktabort(a)=pktabort(a)+1;
        else
           pktcorrect(a)=pktcorrect(a)+1;
        end

        t_modem1(address,cycle+1)=t_modem1(address,cycle)+t_xmit;
        t_modem2(address,cycle+1)=t_xmit;
        d_modem1(address,cycle+1)=d_modem1(address,cycle)+data(m2)*8;
        d_modem2(address,cycle+1)=data(m2)*8;

        DATA=data_set(m2);            %reset data
        OOS=0;                      %reset OOSFLAG correction
        DOOS=0;                      %reset OOSFLAG correction

        counter=counter+1;

    %*************************************************************
    % ENERGY BUDGET: ONLY IN NOISE FREE SIMULATION!!!!!!
    % Instruction: set A=1; T=1; Only for single input parameter
    %*************************************************
%       TT = TT + TOKEN;
%       TI = TI + INFO - (t_delay2+t_delay3);
%       Prcv = (TT)*Wrcv;
%       Pxmt = (TI)*Wxmt;
%       Phr = (Prcv+Pxmt)/number_modems(m6);
%       Crcv = (TI)*Wrcv;
%       Cxmt = (TT)*Wxmt;
%       Chr = (Crcv+Cxmt);
%       energy2b=[Phr; Chr];
    %*************************************************************

      end    % end for loop that determines modem addresses

      t_total(1,cycle+1)=t_total(1,cycle)+sum(t_modem2(:,cycle+1));
      d_total(1,cycle+1)=d_total(1,cycle)+sum(d_modem2(:,cycle+1));
      latency1(1,cycle)=sum(t_modem2(:,cycle+1));

      cycle=cycle+1;

   end       % end while loop - period that network is evaluated

   %*********************************************************************
   % baudrateact = actual baudrate so average actual amount of bits you can
```

```
    %       transfer from peripheral to central considering present settings
    %
    % utilization=t_information/t_total
    %
    % latency = the maximum period that you should have to wait to receive
    %       data from a specific address
    %**************************************************************************

    throughput(a)=(d_total(1,length(d_total))) / (t_total(1,length(t_total)));

    utilization(a)=(d_total(1,length(d_total))/baudrate1(m1)) / (t_total(1,length(t_total)));

    latency(a)=mean(latency1);

    pktabort(a)=100*pktabort(a)/( pktcorrect(a)+pktabort(a) );

  end

util_ave(m1,m2,m3,m4,m5,m6,m7)=mean(utilization);

throughput_ave(m1,m2,m3,m4,m5,m6,m7)=mean(throughput);

latency_ave(m1,m2,m3,m4,m5,m6,m7)=mean(latency);

pktabort_ave(m1,m2,m3,m4,m5,m6,m7)=mean(pktabort);

end
end
end
end
end
end
end

util_ave=squeeze(shiftdim(util_ave));
throughput_ave=squeeze(shiftdim(throughput_ave));
latency_ave=squeeze(shiftdim(latency_ave));
pktabort_ave=squeeze(shiftdim(pktabort_ave));
result2b=[util_ave; throughput_ave; latency_ave; pktabort_ave];
```

The function TOKEN3a.m is used to simulate network strategy T3A.

```
function[result3a, energy3a]=TOKEN3a

clear,

%**************************************************************************
%
```

```
% Initiate values through <ini.m> function
%
%************************************************************************
[number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

for m1=1:length(baudrate1)      % baudrate1
for m2=1:length(data_set)       % data length
for m3=1:length(L2)             % noise
for m4=1:length(maxretries)     % maxretries
for m5=1:length(L_sp)           % subpacket length
for m6=1:length(number_modems)  % number_modems
for m7=1:length(baudrate2)      % baudrate2

   % calculate # subpackets and duration of utility packet

   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));    %number of subpackets
   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;    % transmission time of utility packet

%************************************************************************
%
% Do the simulation a couple of times so you get a good average of the
% metrics you're interested in
%
%************************************************************************

  for a=1:A;        % run simulation 'a' times

% reset all values to zero

   OOS=0;
   DOOS=0;
   SRQ=0;
   DSRQ=0;
   MEMFLAG(1:number_modems(m6),1)=0;
   MEMFLAG(1:number_modems(m6),2)=0;
   RETOKEN=1;
   HUB=1;

   d_total=0;
   t_total=0;
   t_central=0;
   latency1=0;
   latency(a)=0;
   pktabort(a)=0;
   pktcorrect(a)=0;

   d_modem1(1:number_modems(m6),1)=0;
   d_modem2(1:number_modems(m6),1)=0;

   t_modem1(1:number_modems(m6),1)=0;
   t_modem2(1:number_modems(m6),1)=0;
```

```
cycle=1;

%**********************
%energy calc. only !!!
TT=0;
TI=0;
Wrcv=0.5;   %[W]
Wxmt=0.1;   %[W]
%**********************

while t_total<(3600*T)

   for address=1:number_modems(m6);

      %*************************************************************
      % In case MEMFLAG(address,2) is set to 1 the previous packet
      % was out-of-sequence. The packet will be retransmitted at
      % baudrate2. If retransmission fails, the packets gets aborted
      %*************************************************************

      if MEMFLAG(address,2)

         [OOS,DOOS]=mem2(m1,m2,m3,m4,m5,m6,m7);

         MEMFLAG(address,2) = 0;

         if DOOS==0
            pktabort(a)=pktabort(a)+1;
         else
            pktcorrect(a)=pktcorrect(a)+1;
         end

      end

      %*************************************************************
      % In case MEMFLAG(address,1) is set to 1 the previous packet
      % was corrupted. The packet will be retransmitted at
      % baudrate1. If retransmission fails, the packets gets aborted
      %*************************************************************

      if MEMFLAG(address,1)

         [SRQ,DSRQ]=mem1(m1,m2,m3,m4,m5,m6,m7);

         MEMFLAG(address,1) = 0;

         if DSRQ==0
            pktabort(a)=pktabort(a)+1;
         else
            pktcorrect(a)=pktcorrect(a)+1;
         end
```

```
end

%*********************************************************
% info packet
%*********************************************************

[INFO,DATA]=info(m1,m2,m3,m4,m5,m6,m7,RETOKEN);

dice1=rand(1);

%*********************************************************
% utility or info packet corrupted: MEMFLAG(address,1) is
% set to 1 and no data is transmitted
%*********************************************************

dice1=rand(1);

 if ((dice1<L2(m3)) & (dice1>=L3(m3)))

    MEMFLAG(address,1) = 1;     % Subpacket/header corrupted

    DATA = 0;

 end

%*********************************************************
% TOKEN utility packet; if token is corrupted, the central
% node will retransmit token until maxtoken is achieved.
% If maxtoken is needed, TOKENFLAG = 1 and MEMFLAG(address,2)
% is set to 1 which will generate packet out of sequence.
% No data transmission will occur and the packet will be rexmit
% at next cycle
%*********************************************************

[TOKEN,TOKENFLAG]=token(m1,m2,m3,m4,m5,m6,m7,dice1,RETOKEN,HUB);

if TOKENFLAG

   MEMFLAG(address,2) = 1;      % Token corrupted, PKT OOS
   MEMFLAG(address,1) = 0;
   INFO=0; OOS=0;
   DATA=0; DOOS=0; DSRQ=0;

end

%*********************************************************
% add all events
%*********************************************************

t_xmit=TOKEN+INFO+SRQ+OOS;
```

```
        data(m2)=DATA+DOOS+DSRQ;

        if DATA ~= 0
           pktcorrect(a)=pktcorrect(a)+1;
        end

    %*********************************************************
    % ENERGY BUDGET: ONLY IN NOISE FREE SIMULATION!!!!!!
    % Instruction: set A=1; T=1; Only for single input parameter
    %*********************************************
%       TT = TT + TOKEN;
%       TI = TI + INFO - (t_delay2+t_delay3);
%       Prcv = TT*Wrcv;
%       Pxmt = (TT+TI)*Wxmt;
%       Phr = (Prcv+Pxmt)/number_modems(m6);
%       Crcv = (TT+TI)*Wrcv;
%       Cxmt = TT*Wxmt/number_modems(m6);
%       Chr = (Crcv+Cxmt);
%       energy3a=[Phr; Chr];
    %*********************************************************

        t_modem1(address,cycle+1)=t_modem1(address,cycle)+t_xmit;
        t_modem2(address,cycle+1)=t_xmit;
        d_modem1(address,cycle+1)=d_modem1(address,cycle)+data(m2)*8;
        d_modem2(address,cycle+1)=data(m2)*8;

        DATA=data_set(m2);            %reset data
        OOS=0;                      %reset OOSFLAG correction
        DOOS=0;                      %reset OOSFLAG correction
        SRQ=0;
        DSRQ=0;

        counter=counter+1;

    end     % end for loop that determines modem addresses

    %*********************************************************************
    % Add contribution of central hub included in cycle: only 1 token
    % will be generated. If the token from the last peripheral modem to
    % the central modem fails, the central modem knows that it was its
    % turn to transmit so it won;t initiate a new token
    %*********************************************************************

    t_central(1,cycle) = t_wu + t_acq + ...
        (number_modems(m6)*2 + d_ut + d_crc)*8/baudrate2(m7) + t_delay4;

    t_total(1,cycle+1)=t_total(1,cycle)+t_central(1,cycle)...
        +sum(t_modem2(:,cycle+1));
    d_total(1,cycle+1)=d_total(1,cycle)+sum(d_modem2(:,cycle+1));
    latency1(1,cycle)=sum(t_modem2(:,cycle+1));

    cycle=cycle+1;
```

```
    end         % end while loop - period that network is evaluated

    %**********************************************************************
    % baudrateact = actual baudrate so average actual amount of bits you can
    %       transfer from peripheral to central considering present settings
    %
    % utilization=t_information/t_total
    %
    % latency = the maximum period that you should have to wait to receive
    %       data from a specific address
    %**********************************************************************

    throughput(a)=(d_total(1,length(d_total))) / (t_total(1,length(t_total)));

    utilization(a)=(d_total(1,length(d_total))/baudrate1(m1)) / (t_total(1,length(t_total)));

    latency(a)=mean(latency1);

    pktabort(a)=100*pktabort(a)/( pktcorrect(a)+pktabort(a) );

  end

util_ave(m1,m2,m3,m4,m5,m6,m7)=mean(utilization);

throughput_ave(m1,m2,m3,m4,m5,m6,m7)=mean(throughput);

latency_ave(m1,m2,m3,m4,m5,m6,m7)=mean(latency);

pktabort_ave(m1,m2,m3,m4,m5,m6,m7)=mean(pktabort);

end
end
end
end
end
end
end

util_ave=squeeze(shiftdim(util_ave));
throughput_ave=squeeze(shiftdim(throughput_ave));
latency_ave=squeeze(shiftdim(latency_ave));
pktabort_ave=squeeze(shiftdim(pktabort_ave));
result3a=[util_ave; throughput_ave; latency_ave; pktabort_ave];
```

The function poll.m performs polling simulation.

```
function [POLL,POLLFLAG]=poll(m1,m2,m3,m4,m5,m6,m7,dice1,REPOLL)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;

   if ((dice1<L1) & (dice1>=L2(m3)))

      POLLcount=1;

      if REPOLL

         while POLLcount < maxPOLL(m4)
            dice4=rand(1);
            if dice4 < L4(m3)
              POLLcount=POLLcount+1;
            else
              break
            end
         end

      end

      POLL = POLLcount * sum([t_wu t_ut t_delay5 t_delay3]);

      if ( (REPOLL == 0) | (POLLcount == maxPOLL(m4)) )

         POLLFLAG = 1;

      else

         POLLFLAG = 0;

      end

   else

      POLL = sum([t_wu t_ut t_delay1]);

      POLLFLAG = 0;

   end
```

The function token.m performs token simulation.

```
function [TOKEN,TOKENFLAG]=token(m1,m2,m3,m4,m5,m6,m7,dice1,RETOKEN,HUB)

  [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   if HUB
       t_token = t_wu + t_acq + (number_modems(m6)*2 + d_ut + d_crc)*8/baudrate2(m7);
       TOKEN = t_token;
   else
       t_token=t_acq + t_wu + (d_ut+d_crc)*8/baudrate2(m7);
       TOKEN = t_token + t_delay1;
   end

   TOKENFLAG = 0;

   if ((dice1<L1) & (dice1>=L2(m3)))

      TOKENcount=1;

      if RETOKEN == 0
         maxTOKEN(m4) = 1;
      end

      while TOKENcount <= maxTOKEN(m4)
           dice4=rand(1);
           if dice4 < L4(m3)
              TOKENcount=TOKENcount+1;
           else
              break
           end
      end

      if ( TOKENcount > maxTOKEN(m4) )

         TOKENcount = maxTOKEN(m4);

         TOKENFLAG = 1;

      end

         TOKEN = TOKEN + TOKENcount * sum([t_token t_delay5 t_delay3]);

   end
```

The function info.m generates data packets.

```
function [INFO,DATA]=info(m1,m2,m3,m4,m5,m6,m7,type)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;

   if type == 0
      N_sp(m5)=1;
   else
      N_sp(m5)=ceil(data_set(m2)/L_sp(m5));
   end

   INFO = sum([t_wu t_ut (data_set(m2)+N_sp(m5)*d_crc+d_nw)*8/baudrate1(m1)...
               t_delay2 t_delay3]);

   DATA = data_set(m2);
```

The function oos.m creates packets that are retransmitted upon packet-out-of-sequence situation.

```
function [OOS,DOOS,OOSFLAG]=oos(m1,m2,m3,m4,m5,m6,m7,OOSFLAG,REPOLL)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;
   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));

   dice1=(L3(m3)+L2(m3))/2;

   [SRQ,DATA,OOSFLAG]= srq1(m1,m2,m3,m4,m5,m6,m7,dice1,OOSFLAG,REPOLL);

   if DATA == 0
            ACK = 0;
   else
      [ACK]=ack(m1,m2,m3,m4,m5,m6,m7,OOSFLAG);
    end

   OOS = SRQ + ACK;

   DOOS = DATA;
```

The function srq1.m determines what kind of unsuccessful initial transmission occurs and how the retransmission is performed.

```
function [SRQ,DATA,OOSFLAG]= srq1(m1,m2,m3,m4,m5,m6,m7,dice1,OOSFLAG,REPOLL)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;
   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));

   if ((dice1<L2(m3)) & (dice1>=L3(m3)))

            dice2=rand(1);

             for i=1:(N_sp(m5))

               if ( (dice2<(i/N_sp(m5))) & (dice2>=(i-1)/N_sp(m5)) )

                  dice3=rand(1);

            %*********************************************************
            % header corrupted (causes packet out of sequence)
            % subpacket corrupted + SRQ corrupted
            %*********************************************************

                   if (dice3<L23(m3))

                     if REPOLL

                        i=N_sp(m5);

                        [SRQ,DATA] = srq2(m1,m2,m3,m4,m5,m6,m7,i,OOSFLAG);

                        OOSFLAG = 0;

                     else

                        SRQ = sum([t_delay4 (3+15*i)*maxretries(m4)]);

                        DATA = 0;

                        OOSFLAG = 1;

                     end

            %*********************************************************
```

```
            % subpackets corrupted (SRQ)
            %*********************************************************

                  else

                     [SRQ,DATA]=srq2(m1,m2,m3,m4,m5,m6,m7,i,OOSFLAG);

                     OOSFLAG = 0;

                  end

                  break

               end

            end

   else

      DATA = data_set(m2);

      OOSFLAG = 0;

      SRQ = 0;
   end
```

The function srq2.m creates packets that are retransmitted upon unsuccessful initial transmission.

```
function [SRQ,DATA]=srq2(m1,m2,m3,m4,m5,m6,m7,i,OOSFLAG)


   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;
   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));

   srqcount=1;

   if OOSFLAG
      baudrate = baudrate2(m7);
   else
      baudrate = baudrate1(m1);
   end
```

```
while srqcount <= maxretries(m4)
     dice4=rand(1);

     if dice4 < L4(m3)
       srqcount=srqcount+1;
     else
       break
     end
end

% in case of more than max SRQ, the packet gets aborted

if srqcount > maxretries(m4)
   srqcount=maxretries(m4);
   DATA=0;
else

   DATA=data_set(m2);

end

% SRQ calculation is not really 1 on 1 because multiple retransmissions
% (srqcount) do not necessarily have to be the same length

   SRQ=sum([ srqcount * sum([t_delay4 t_ut t_delay1 t_wu t_ut...
      i*L_sp(m5)*8/baudrate] )]);
```

The function mem1.m can be compared to srq1.m and handles retransmissions in case of T3A.

```
function [SRQ,DSRQ]=mem1(m1,m2,m3,m4,m5,m6,m7)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;
   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));

   dice2=rand(1);

   for i=1:(N_sp(m5))

     if ( (dice2<(i/N_sp(m5))) & (dice2>=(i-1)/N_sp(m5)) )
```

```
        SRQ = sum([t_delay4 t_ut i*L_sp(m5)*8/baudrate1(m1)]);

        break

      end

    end

    dice1=rand(1);

    if ((dice1<L2(m3)) & (dice1>=L3(m3)))

        DSRQ = 0;

    else

        DSRQ = data_set(m2);

    end
```

The function mem2.m handles retransmissions in case packet-out-of-sequence for T3A.

```
function [OOS,DOOS]=mem2(m1,m2,m3,m4,m5,m6,m7)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
    t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
    L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

    t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;
    N_sp(m5)=ceil(data_set(m2)/L_sp(m5));

    OOS = sum([t_delay4 t_ut ...
                (data_set(m2)+N_sp(m5)*d_crc+d_nw)*8/baudrate2(m7)]);

    dice1=rand(1);

    if ((dice1<L2) & (dice1>=L3(m3)))

        DOOS = 0;

    else

        DOOS = data_set(m2);

    end
```

The function ack.m handles ACK messages for packet-out-of-sequence situations.

```
function [ACK]=ack(m1,m2,m3,m4,m5,m6,m7,OOSFLAG)

   [number_modems t_wu t_acq d_ut d_crc d_nw t_delay1 t_delay2...
   t_delay3 t_delay4 t_delay5 data_set baudrate1 baudrate2 maxretries...
   L_sp A T alfa L1 L2 L3 L23 L4 maxCTS maxACK maxPOLL maxTOKEN]=INI;

   t_ut=(d_ut+d_crc)*8/baudrate2(m7)+t_acq;
   N_sp(m5)=ceil(data_set(m2)/L_sp(m5));

    if OOSFLAG
       baudrate = baudrate2(m7);
    else
       baudrate = baudrate1(m1);
    end

       ACK = t_delay4 + t_ut;

       diceACK=rand(1);

    if ((diceACK<L1) & (diceACK>= (L2(m3)+L1)/2 ))

        ACKcount=1;

        while ACKcount < maxACK(m4)
            dice4=rand(1);
            if dice4 < L4(m3)
               ACKcount=ACKcount+1;
            else
               break
            end
        end

        ACK = ACKcount*sum([ACK t_delay2 (data_set(m2)+N_sp(m5)*d_crc+d_nw)*8/baudrate]);

     end
```

LIST OF REFERENCES

[1] J. A. Rice, "Seaweb Underwater Networks," Proceedings Acoustics 2005, Australian Acoustical Society Annual Conference, Busselton: Western Australia, Nov. 2006.

[2] M. Stojanovic, "On the Relationship between Capacity and Distance in an Underwater Acoustic Communication Channel," Proceedings First ACM International Workshop on Underwater Networks (WUWNet), Los Angeles, Sep. 2006.

[3] A. F. Harris III, D. G. B. Meneghetti, M. Zorzi, "Maximizing Channel Utilization for Underwater Acoustic Links," Proceedings IEEE Oceans, Aberdeen, Jun. 2007.

[4] D. E. Lucani, M. Medard, M. Stojanovic, "Network Coding Schemes for Underwater Networks," Proceedings Second ACM International Workshop on Underwater Networks (WUWNet), Montreal, Sep. 2007.

[5] R. F. W. Coates, Underwater Acoustic Systems, New York: Halsted Press, 1989.

[6] R. Urick, Principles of Underwater Sound, 3rd ed., Los Altos: Peninsula Publishing, 1983.

[7] L. E. Kinsler, A. R. Frey, A. B. Coppens, J. V. Sanders, Fundamentals of Acoustics, 4th ed., New York: John Wiley & Sons Ltd., 2000.

[8] L. J. Ziomek, Fundamentals of Acoustic Field Theory and Space-Time Signal Processing, Boca Raton: CRC Press, 1995.

[9] L. J. Ziomek, Course Notes for EC3450 Fundamentals of Ocean Acoustics, Naval Postgraduate School, Monterey, CA, 2007.

[10] C. S. Clay, H. Medwin, Acoustical Oceanography: Principles and Applications, Wiley, New York, 1977.

[11] J. T. Hansen, "Link Budget Analysis for Undersea Acoustic Signaling," MS thesis Naval Postgraduate School, Jun. 2002.

[12] R. E. Francois and G. R. Garrison, "Sound absorption based on ocean measurements: Part I: Pure water and magnesium sulfate contributions," Journal of the Acoustical Society of America, vol. 72, no. 3, 1982, 896–907.

[13] R. E. Francois and G. R. Garrison, "Sound absorption based on ocean measurements: Part II: Boric acid contribution and equation for total absorption," Journal of the Acoustical Society of America, vol. 72, no. 6, 1982, 1879–1890.

[14] W. H. Thorp, “Analytic Description of Low Frequency Attenuation Coefficient,” Journal of the Acoustical Society of America, vol. 42, 1967, 270–271.

[15] F. H. Fisher and V. P. Simmons, “Sound Absorption in Sea Water,” Journal of the Acoustical Society of America, vol. 62, 1977, 558–564.

[16] M. Schulkin, H. W. Marsch, “Sound Absorption in Sea Water,” Journal of the Acoustical Society of America, vol. 34, 1962, 864–865.

[17] G. M. Wenz, “Acoustic Ambient Noise in the Ocean: Spectra and Sources,” Journal of the Acoustical Society of America, vol. 34, no. 23, 1962, 1936–1956.

[18] M. Stojanovic, “Underwater Acoustic Communication,” Wiley Encyclopedia of Electrical and Electronics Engineering, Dec. 1999.

[19] I. F. Akyildiz, D. Pompili and T. Melodia, “Underwater Acoustic Sensor Networks: Research Challenges,” Ad Hoc Networks (Elsevier), vol. 3, no. 3, 257–279, May 2005.

[20] H. P. E. Stern, S. A. Mahmoud, L. E. Stern, Communication Systems Analysis and Design, Upper Saddle River, New Jersey: Pearson Education, Inc., 2004.

[21] J. G. Proakis, M. Salehi, Communication Systems Engineering, 2nd ed., Upper Saddle River, New Jersey: Pearson Prentice Hall, 2002.

[22] W. Stallings, Data and Computer Communications, 8th ed., Upper Saddle River, New Jersey: Pearson Education, Inc., 2007.

[23] J. G. Proakis, Digital Communications, 4th ed., New York: McGraw-Hill, 2001.

[24] B. Kim, I. Lu, “Parameter Study of OFDM Underwater Communications System,” OCEANS 2000 MTS/IEEE Conference and Exhibition, Vol. 2, 1251–1255, 11-14 Sep. 2000.

[25] S. Coatelan, A. Glavieux, “Design and Test of a Multi-carrier Transmission System on the Shallow Water Acoustic Channel,” OCEANS ’94 Proceedings, Vol. 3, III-472 – III-477, 1994.

[26] G. T. L. Pitman III, "Simulation of an Orthogonal Frequency Division Multiplexing Based Underwater Communication System using a Physics Based Model for the Underwater Acoustic Sound Channel," MS thesis Naval Postgraduate School, Sep. 2001.

[27] L. W. Couch, Digital and Analog Communication Systems, 7th ed., Upper Saddle River, New Jersey: Pearson Education, Inc., 2007.

[28] O. B. Wilson, Introduction to Theory and Design of Sonar Transducers, 1988 ed., Los Altos: Peninsula Publishing, 1991.

[29] Sensor Technology Limited. Sensortech Products/Acoustic Transducers/Free Flooded Rings, retrieved Oct. 3, 2007 from Sensor Technology Limited website: www.sensortech.ca/site/index.cfm?DSP=Page&ID=99.

[30] ITC International Transducer Corporation. Underwater Transducers, retrieved Oct. 10, 2007 from International Transducer Corporation website: www.itc-transducers.com/html/underwater_transducers.html.

[31] E. Kuntsal, W. A. Bunker, "Guidelines for Specifying Underwater Electroacoustic Transducers," Revised version of paper presented at UDT '92 Conference, June 1992, London, England, retrieved Oct. 11, 2007 from International Transducer Cooperation website: www.itc-transducers.com/pdf/guidespec.pdf.

[32] D. Halliday, R. Resnick, J. Walker, Fundamentals of Physics, 7th ed. (extended), New York: John Wiley and Sons, Inc., 2005.

[33] J. A. Rice, R. C. Shockley, "Battery-Energy Estimates for Telesonar Modems in a Notional Undersea Network," Proceedings MTS Ocean Community Conference, vol. 2, 1007–1015, Baltimore, Nov. 1998.

[34] P. Casari, S. Marella, M. Zorzi, "A Comparison of Muliple Access Techniques in Clustered Underwater Acoustic Networks," Proceedings IEEE Oceans, Aberdeen, Jun. 2007.

[35] E. M. Sozer, M. Stojanovic, J. G. Proakis, "Underwater Acoustic Networks," IEEE Journal of Oceanic Engineering, vol. 25, no. 1, Jan. 2000.

[36] M. Stojanovic, "Frequency Reuse Underwater: Capacity of an Acoustic Cellular Network", Proceedings Second ACM International Workshop on Underwater Networks (WUWNet), Montreal, Sep. 2007.

[37] M. Stojanovic, L. Freitag, "Multichannel Detection for Wideband Underwater Acoustic CDMA Communications," IEEE Journal of Oceanic Engineering, vol. 31, no. 3, Jul. 2006.

[38] J. M. Kalscheuer, “A Selective Automatic Repeat Request Protocol for Undersea Acoustic Links,” MS thesis Naval Postgraduate School, Jun. 2004.

[39] C. L.Fletcher, J. A. Rice, R. K. Creber, “Undersea Acoustic Network Operations through a Database-Oriented Server/Client Interface,” Proceedings IEEE Oceans 2001 Conference, 2071–2075, Nov. 2001.

[40] J. K. Wilson, “Maritime Surveillance using a Wideband Hydrophon,” MS thesis Naval Postgraduate School, Sep. 2007.

[41] J. C. Torres, “Modeling of High-Frequency Acoustic Propagation in Shallow Water,” MS thesis Naval Postgraduate School, Jun. 2007.

[42] N. Metropolis, S. Ulam, "The Monte Carlo Method," Journal of the American Statistical Association, vol. 44, pp. 335-341, Sep. 1949.

[43] Monterey Bay 2006 Field Experiments. Underwater Persistent Surveillance, retrieved Nov. 5, 2007 from Monterey Bay Aquarium Research Institute (MBARI) website: http://www.mbari.org/MB2006/UPS/mb2006-ups-links.htm.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Joseph A. Rice
   Department of Physics
   Naval Postgraduate School
   Monterey, California

4. Prof. Lawrence J. Ziomek
   Department of Electrical Engineering
   Naval Postgraduate School
   Monterey, California

5. Prof. Kevin B. Smith
   Department of Physics
   Naval Postgraduate School
   Monterey, California

6. RADM (Ret) Ray Jones
   Naval Postgraduate School
   Monterey, California

7. RADM (Ret) Rick Williams
   Naval Postgraduate School
   Monterey, California

8. Prof. Don Brutzman
   Naval Postgraduate School
   Monterey, California

9. Wendy Walsh
   Naval Postgraduate School
   Monterey, California

10. CDR RNLN L.R. Jacobs
    SWT Division, Underwater Technology
    Defense Materiel Organization
    Gravenhage, The Netherlands

11. Vincent van Leijen
Faculty of Military Sciences
Netherlands Defense Academy
Den Helder, The Netherlands

12. CAPT RNLN Victor C. Windt
The Royal Netherlands Embassy
Washington DC

13. David Du
National Science Foundation
Arlington, Virginia

14. Bill Marn
SPAWAR Systems Center
San Diego, California

15. Chris Fletcher
SPAWAR Systems Center
San Diego, California

16. Bob Creber
SPAWAR Systems Center
San Diego, California

17. Doug Grimett
SPAWAR Systems Center
San Diego, California

18. Keyko McDonald
SPAWAR Systems Center
San Diego, California

19. Dana Hesse
ONR
Arlington, Virginia

20. Dave Johnson
ONR
Arlington, Virginia

21. Tom Drake
ONR
Arlington, Virginia

22. Ralph Wachter
ONR
Arlington, Virginia

23. Jody Wood-Putnam
Naval Surface Warfare Center
Panaman City, Florida

24. Phil Bernstein
Naval Surface Warfare Center
Panama City, Florida

25. LT RNLN Bjørn E. A. Kerstens
Royal Netherlands Navy
Enkhuizen, The Netherlands